Издательство «Знание»

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Пользователям профессионального компьютера

1990

11

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

11/1990

Издается
ежемесячно
с 1988 г.

ПОЛЬЗОВАТЕЛЯМ
ПРОФЕССИОНАЛЬНОГО
КОМПЬЮТЕРА

В номере:

С.И.Кляус
Пакет NORTON COMMANDER

В.В.Слободчук
Пакет "VITAMIN C"

РУБРИКИ
Нам пишут

Издательство
"Знание"
Москва
1990

ББК 32.85
П53

Авторы выпуска

КЛЯУС Сергей Витальевич — научный сотрудник, занимается разработкой программного обеспечения АСУ, СУБД и систем передачи данных.

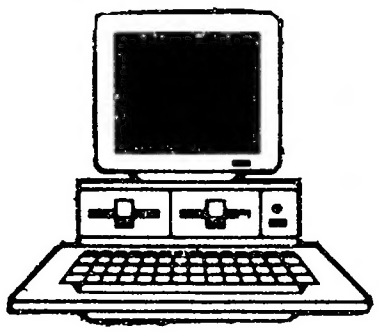
СЛОБОДЧУК Вячеслав Викторович — научный сотрудник, занимается исследованием и разработкой локальных вычислительных сетей.

ЖАРИКОВ Л.Н., СУХОРУКОВ Ф.Д. — программисты.

РЕДАКТОР Б. М. ВАСИЛЬЕВ

ISBN 5-07-001610-5

© Издательство "Знание", 1990 г.



Данная статья ставит своей целью ознакомить пользователей персональных ЭВМ с файловым монитором "Norton commander". В статье приводится достаточно полная информация о практических приемах работы с этим пакетом, в ней раскрыты основные возможности и особенности версии 2.0.

С.В.Кляус

Пакет NORTON COMMANDER

Одной из характерных черт развития нашего общества в настоящий момент является повсеместное внедрение средств электронно-вычислительной техники. Особую интенсивность этот процесс приобрел с появлением персональных ЭВМ. Трудно представить себе какую-либо сферу деятельности человека, где не было бы попыток внедрения ПЭВМ. Эксплуатация ПЭВМ, в свою очередь, не мыслима без эксплуатации программного обеспечения. И основной среди бесконечного множества программных продуктов является, конечно, операционная система.

С увеличением парка персональных ЭВМ в нашей стране наибольшую популярность завоевывают ПЭВМ типа IBM AT/XT и их клоны из различных видов зарубежных машин и ППЭВМ типа ЕС 1840/41 из отечественных машин.

Из операционных систем, подходящих для этих ЭВМ, наибольшее распространение получила дисковая операционная система MS-DOS. Основные команды MS-DOS имеют простой синтаксис и не требуют особых усилий для их запоминания. Однако при увеличении количества дискет и файлов, находящихся в эксплуатации у пользователя, затруднения в работе с ПЭВМ возрастают, так как требуется запоминание названий каталогов, файлов, их содержимого и т.д. В такой ситуации работа с "голой" ДОС становится весьма нудным занятием, которое требует терпения и значительных затрат времени. Для упрощения этих действий создаются специальные инструментальные средства, своеобразные надстройки над ДОС, позволяющие выполнять большинство команд без набора их мнемокода и опций.

Одним из первых инструментальных средств, с которым сталкивается пользователь персональной ЭВМ, часто является пакет программ "Norton commander", в дальнейшем NC. Популярность этого инструмента необычайна. Да это и понятно, Обладая "дружественным" интерфейсом, он чрезвычайно легок в освоении и создает массу удобств при работе с файлами и дискетами.

Программист, использующий NC, не выходя из него, может разрабатывать и запускать свои программы и пакеты; предварительно запоминать и выполнять команды MS-DOS, что в дальнейшем позволяет избегать их повторных наборов и избавляет от рутинной работы. NC, таким образом, играет роль надстройки над MS-DOS. Подобные программы часто называют файловыми мониторами. 1*) (такое обозначение расшифровывается в примечаниях).

После завершения работы прикладной программы или команды MS-DOS, выбранной пользователем, происходит возвращение к функционированию NC.

Внешний вид (изображение на экране монитора) работающего NC может иметь вид, изображенный на рис.1. 2*) Полноэкранная форма включает в себя четыре основных компонента:

- левую панель;
- правую панель;
- командную строку (расположена под панелями и начинается с указания активного дисковода);
- строку-подсказку (расположена под командной строкой, в ней приведен перечень используемых функциональных клавиш и кратко указаны соответствующие им действия).

```
                      D:\                                             C:\
 Name              Size      Date      Time       Name              Size      Date      Time
ACAD             ►SUB-DIR◄  5-22-90   7:52a     3D                ►SUB-DIR◄  1-10-90  12:51p
DRIVERS          ►SUB-DIR◄  6-22-90  12:29p     AAQW              ►SUB-DIR◄  1-30-90   3:22p
GAMES            ►SUB-DIR◄  1-18-90   7:15p     ADVERT            ►SUB-DIR◄  3-26-90  11:36a
KAR              ►SUB-DIR◄  5-11-90  11:39a     ALSY              ►SUB-DIR◄  2-03-90  12:02p
LESHA            ►SUB-DIR◄ 10-09-90   5:35a     AVIR              ►SUB-DIR◄  2-12-90   9:08p
MODE             ►SUB-DIR◄  6-22-90  12:29p     BETA              ►SUB-DIR◄  1-12-90   2:51p
PRINT            ►SUB-DIR◄  5-10-90   7:11a     BIG               ►SUB-DIR◄  2-05-90   3:28p
TUTORPC          ►SUB-DIR◄  6-19-90  11:54a     CREC              ►SUB-DIR◄  7-30-90   6:23a
TYPESET          ►SUB-DIR◄  9-26-90   5:28a     DOS               ►SUB-DIR◄  1-05-90   8:38p
VENTURA          ►SUB-DIR◄  9-26-90   5:28a     JINN              ►SUB-DIR◄  1-18-90   2:45p
WORD5            ►SUB-DIR◄  8-03-90   5:55a     MARI              ►SUB-DIR◄  5-24-90   7:00a
igor      chp          746  9-12-90   6:04a     MOUSE1            ►SUB-DIR◄  1-22-90   4:07p
igor      cif          128  9-12-90   6:03a     NEWS              ►SUB-DIR◄  1-10-90  12:42p
karina    $hp          635  9-05-90  10:06a     NORTON            ►SUB-DIR◄  1-05-90   8:32p
karina    chp          635  9-05-90  10:28a     PCSHIELD          ►SUB-DIR◄  4-18-90   8:51a
karina    cif          128  9-05-90  10:28a     QUATTRO           ►SUB-DIR◄  4-11-90   8:33a
lp        bas           32 10-12-90  11:34a     SHOW              ►SUB-DIR◄  4-03-90  10:44a
treeinfo  ncd          683 10-09-90   7:03a     TAN               ►SUB-DIR◄  9-17-90   5:33a

ACAD             ►SUB-DIR◄  5-22-90   7:52a     3D                ►SUB-DIR◄  1-10-90  12:51p
D:\>
1Help  2User  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9Menu  10Quit
```

Рис.1

Основные операции, выполняемые NC, приведены в строке-подсказке, и при нажатии соответствующих функциональных клавиш позволяют выполнить следующие действия:

1Help — F1 или Ф1 (для ПП ЭВМ типа ЕС1840, ЕС1841), выводит на экран краткую инструкцию по работе с NC;

2User — F2 (Ф2), активизирует меню процедур, предварительно созданных пользователем ; один из возможных вариантов приведен на рис.2;

```
                         C:\                                            D:\
   Name           Name             Name            Name            Name              Name
3D            WORD            mouse     sys     ACAD            treeinfo ncd
AAQW          Frecover idx    nc        mnu     DRIVERS         vpprof   bat
ADVERT        Io       sys    razd5     doc     GAMES
ALSY          Msdos    sys
AVIR          autoexec bat         ┌──── User Menu ────┐
BETA          autoexec old         │ P     PCTools     │
BIG           command  com         │ E     EDITOR      │
CREC          config   old         │ F     Format 730K │
DOS           config   sys         │ N     NARC        │
JINN          frecover dat         │ T     TIME        │
MARI          hpemdsc  sys         │ V     VERSION     │
MOUSE1        hpemmgr  sys         │ D     DATE        │
NEWS          kbd      exe         └───────────────────┘
NORTON        letter   bak
PCSHIELD      letter   doc                      karina   chp
QUATTRO       letter1  bak                      karina   cif
SHOW          letter1  doc                      lp       bas
TAN           model    bat                      nc       mnu

3D            ►SUB-DIR◄  1-10-90 12:51p         ACAD            ►SUB-DIR◄  5-22-90   7:52a
D:\>
1Help  2User  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9Menu  10Quit
```

Рис.2

3View — F3 (Ф3), позволяет осуществить просмотр выбранного файла;

4Edit — F4 (Ф4), дает возможность отредактировать файл размером до 31 кбайта. При большем объеме редактирование невозможно, но допускается переход к просмотру;

5Copy — F5 (Ф5), позволяет копировать файл (группу помеченных файлов) как с одной дискеты на другую, так и в пределах одного директория, но в последнем случае необходимо различие в именах копируемого и нового файлов;

6RenMov — F6 (Ф6), используется для перемещения или переименования файла (группы помеченных файлов);

7Mkdir — F7 (Ф7), предлагает создать в текущем директории новый подкаталог;

8Delete — F8 (Ф8), приводит к уничтожению заранее выбранного файла (группы помеченных файлов после подтверждения);

9Menu — F9 (Ф9), активизирует меню NC, которое выводится затем в виде верхней строки — подсказки;

10Quit — F10 (Ф10), прекращает работу NC и освобождает память.

Рассмотрим выполнение некоторых операций более подробно.

I. При нажатии функциональной клавиши F1 (Ф1) на экран монитора выводится таблица “Help“, которая содержит краткий перечень команд и опций NC (рис.3). Таблица напоминает, что NC создан Джоном Сочей (John Socha) и имеет в своем составе пять разделов:

1. Раздел “CURSOR MOVEMENT“ информирует о возможностях управления перемещением курсора:

- “Char left“ — на одну позицию влево достигается одновременным нажатием клавиш “Ctrl-S“ (“УПР-S“);
- “Char right“ — на одну позицию вправо нажатием “Ctrl-D“ (“УПР-D“);
- “Word left“ — на одно слово влево “Ctrl-A“ (“УПР-A“);
- “Word right“ — на одно слово вправо “Ctrl-F“ (“УПР-F“).

2. Раздел “DELETE“ информирует о возможностях удаления символов и слов:

Рис.3

- “Char left“ — один символ слева от курсора удаляется одновременным нажатием клавиш “Ctrl-H“ или “BS“ (“УПР-H“ или “←“);
- “Word left“ — слово слева от курсора “Ctrl-W“ или “Ctrl-BS“ (“УПР-W“ или “УПР-← “); .
- “Under cursor“ — символ, помеченный курсором, удаляется одновременным нажатием “Ctrl-G“ или “Del“ (“УПР-G“ или “УДЛ“);
- “Word right“ — удаление слова справа от курсора “Ctrl-T“ (“УПР-T“);
- “Line“ — удаление содержимого командной строки “Ctrl-Y“ (“УПР-Y“).

3. Раздел “PANELS“ дает информацию о возможностях управления панелями:
- “Switch panels“ — переключение активной панели — “Ctrl-I“ или ‹Tab› (“УПР-I“ или “⇄“);
- “Panels on/off“ — включение/выключение панелей — “Ctrl-O“ (“УПР-O“);
- “Panels 1 or 2“ — выключение/включение неактивной в текущий момент панели — “Ctrl-P“ (“УПР-P“);
- “Swap panels“ — перестановка панелей местами — “Ctrl-U“ (“УПР-U“);
- “Toggle status“ — переключение статуса активной панели “Ctrl-L“ (“УПР-L“);
- “Toggle bar menu“ — переключение активности линейки меню “Ctrl-B“ (“УПР-B“).

4. Раздел “OTHER“ напоминает о некоторых других возможностях NC:
- “cd..“ — выйти из подкаталога — “Ctrl-PgUp“ (“УПР-⇞“);
- “cd ‹name›“ — войти в подкаталог — “Ctrl-PgDn“ (“УПР-⇟“);
- “cd\“ — возвратиться в главный каталог — “Ctrl-\“ (“УПР-\“);
- “Reread directory“ — повторное чтение каталога активного директория — “Ctrl-R“ (“УПР-R“);
- “Select/Unselect“ — пометить файл/снять пометку с файла — ‹Ins› (“ВСТ“);
- “Previous command“ — поместить в командную строку предыдущую команду — “Ctrl-E“ (“УПР-E“);
- “Next command“ — подготовить командную строку для набора следующей команды — “Ctrl-X“ (“УПР-X“).

5. Последний раздел напоминает о том, что клавишами “ + “ и “-“ можно производить пометку или отмену метки групп файлов в соответствии с какими-либо признаками.

Отмена вывода таблицы помощи на экран осуществляется клавишей “ESC“ (“Ключ“) или F10 (“Ф10“).

К приведенному в подсказке можно добавить, что нажатие: “Ctrl-Y(“УПР-Y“) приводит к удалению строки, на которой находится курсор;

“Ctrl-E“(“УПР-E“) или “↑“ приводит к перемещению курсора на строку вверх;

“Ctrl-X“(“УПР-X“) или “↓“ — на строку вниз;

“Ctrl-C“(“УПР-C“) или RgDn выводит следующую страницу экранного изображения;

“Ctrl-R“(“УПР-R“) или PgUp — возврат предыдущей страницы;

“Home“(“↖“) — перемещает курсор в начало строки;

“Ctrl-Home“(“УПР-↖“) — в начало файла;

“Ctrl-K“(“УПР-K“) или “End“(“КОН“) — в конец строки;

“Ctrl-End“(“УПР-КОН“) — в конец файла.

II. При нажатии функциональной клавиши F4 (Ф4) происходит активизация внутреннего редактора, заранее определенного пользователем. Это дает возможность откорректировать предварительно выбранный файл. Следует предупредить о необходимости аккуратного использования этой функции, так как неосторож-

ная коррекция может привести к порче файла, что в ряде случаев (например, файлы типа EXE или COM) может быть необратимым.

Первоначальное создание файла производится следующим образом:

- одновременно нажать клавиши “Shift-F4“ (“↑-Ф4“);
- указать имя, под которым новый файл будет записан на дискету и внесен в текущий каталог и нажать клавишу “Enter“(ВВОД“);
- после появления реплики выбрать ответ “New-file“(“Новый файл“).

В активном режиме редактор может использовать функциональные клавиши:

1Help — F1(“Ф1“), вывод на экран таблицы помощи;

2Save — F2(“Ф2“), запись корректируемого файла на дискету без выхода из режима коррекции;

7Search — F7(“Ф7“), поиск в тексте файла символьной цепочки (шаблон для поиска задается в рамке, появляющейся на экране после нажатия функциональной клавиши);

1Quit — F10(“Ф10“), выход из режима коррекции.

В процессе редактирования на верхней строке экрана приводится следующая информация:

Edit: — режим редактирования;

Имя корректируемого файла;

Line — номер строки, на которой находится курсор;

Col — номер колонки, на которой находится курсор;

Free — количество свободной памяти, доступной для наращивания файла;

Код символа, помеченного курсором. Если курсор находится в конце строки, вместо кода приводится аббревиатура EOL (end of line)7

Следует отметить полезную особенность редактора NC. Если пользователь перед выходом по какой-либо причине не записал корректируемый файл на дискету, то NC потребует от него выбора:

Save — выход с записью откорректированного файла;

Don´t save — выход без записи;

Continue editing — продолжение редактирования.

III. При нажатии функциональной клавиши F9 (Ф9) активизируется меню NC, что видно по появлению в верхней части панелей еще одной строки-подсказки (рис.4). Меню NC имеет в своем составе следующие пункты:

- “Left“ и “Right“ — управление выводом информации на панели, соответственно левую и правую;
- “Files“ — выполнение действий с файлами;
- “Commands“ — выполнение команд NC;
- “Options“ — выполнение опций NC.

Выбор правой или левой панели осуществляется с помощью соответствующего управляющего слова, что приводит к активизации меню управления панелью (рис.5), состоящее из трех групп команд. Первая группа команд управляет видом информации, выводимой на панель. Она включает в себя следующие команды:

“Brief“ — сведения о файлах и подкаталогах будут выводится в кратком виде (имя и расширение) в три столбца;

“Full“ — сведения будут выводиться с указанием имени и расширения (“Name“), размера в байтах (“Size“), даты (“Date“) и времени (“Time“) записи файла на дискету. Формат даты и времени можно изменять посредством задания требуемого значения в операторе “country = “ конфигурационного (config.sys) файла;

“Info“ — на активную панель будет выведена информация о свободных объемах оперативной памяти ЭВМ и дискеты;

“Tree“ — на активную панель будет выведено “дерево“ подкаталогов активного дисковода со схемой их путей.

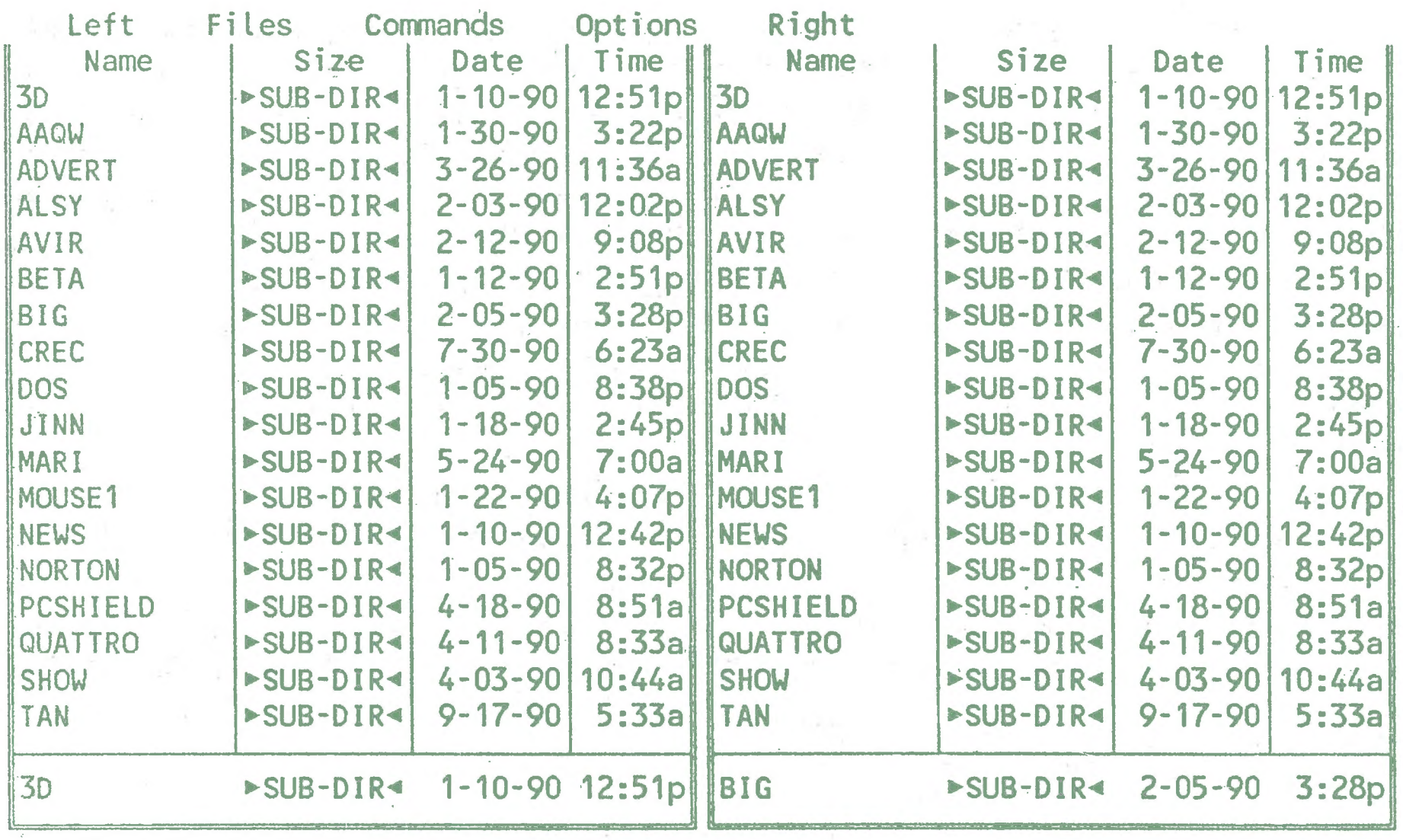


Рис.4

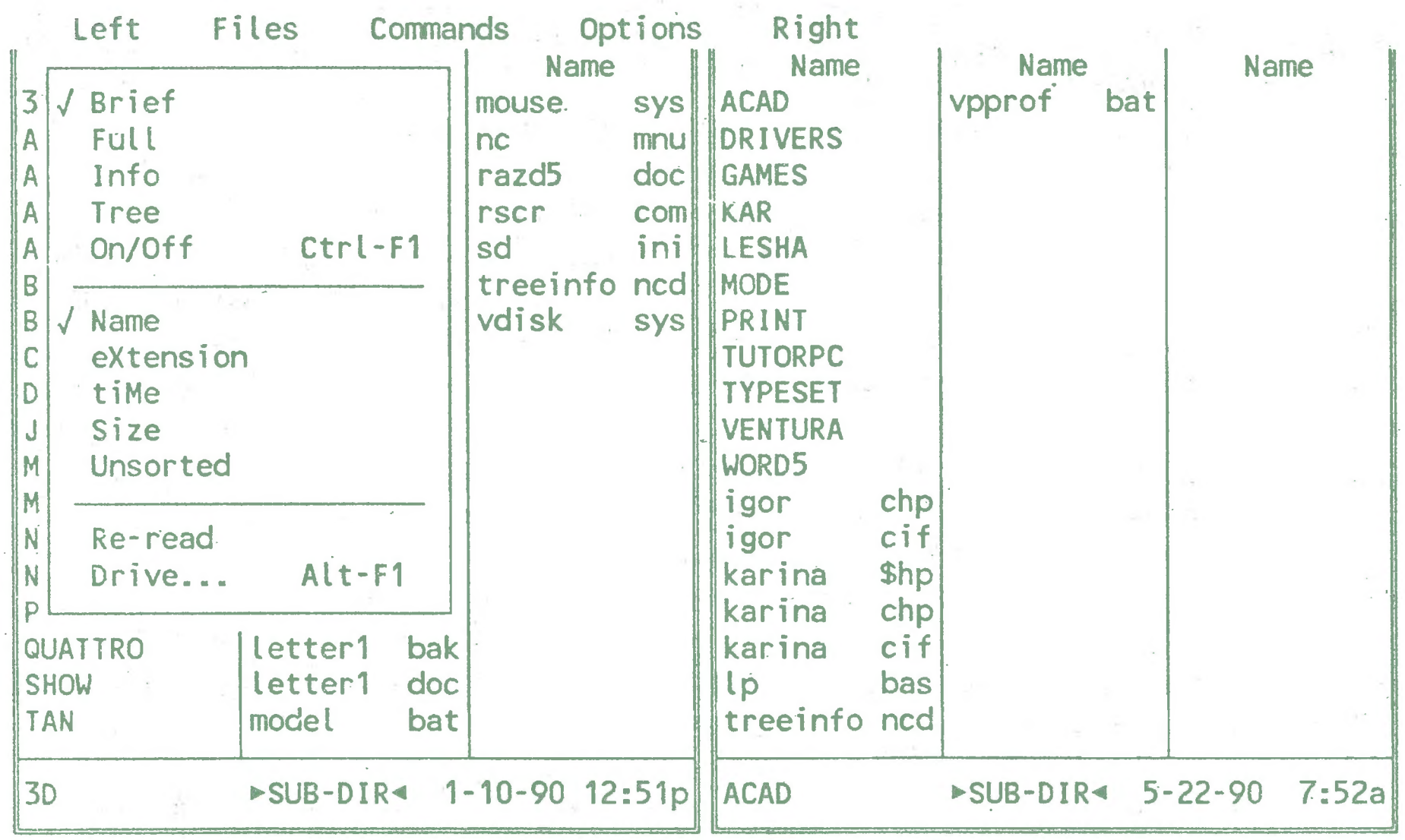


Рис.5

Одновременное нажатие клавиш “Ctrl-L“(“УПР-L“) вызывает переход неаивной панели от состояния “Info“ к состоянию “Brief“ (“Full“ или “Tree“) и наоборот.

“On/Off“ — включает/отключает вывод изображения панели на экране.

Вторая группа команд управляет сортировкой имен подкаталогов и файлов,

выводимых на панель. Она включает в себя следующие команды: “Name“ — вывод будет осуществляться в алфавитном порядке имен; “eXtention“ — в алфавитном порядке расширений;

“tiMe“ — по времени записи на дискету, в первую очередь будут выводиться более свежие файлы;

“Size“ — по размерам файлов, сначала будут выводиться большие файлы.

В случае одинакового времени записи или одинаковых размеров файлов сортировка между ними будет осуществляться по алфавиту.

“Unsorted“ — вывод будет осуществляться в соответствии с физическим расположением файлов.

Третья группа включает в себя команды повторного чтения директория логического дисковода (“Re-read“) и выбора другого дисковода (“Drive...“) с чтением соответствующего каталога. 3*).

Меню выполнения действий над файлами (рис.6) активизируется с помощью выбора управляющего слова “Files“. Нетрудно заметить, что содержимое этого меню идентично содержимому нижней строки-подсказки.

Выбор управляющего слова “Commands“ верхней строки-подсказки дает возможность работы в меню команд NC (рис.7):

1. “NCD tree“ выдает на экран дерево подкаталогов (рис.8) и позволяет войти в любой из них.4*) Выбор требуемого подкаталога можно осуществить клавишами управления курсором, набором имени подкаталога в поле “Speed search“ или одновременным нажатием клавиш “Ctrl-Enter“ (“УПР-ВВОД“). Если при использовании клавиш “Ctrl-Enter“ в поле “Speed search“ будет указано начало имени, которое является общим для нескольких подкаталогов, то перебор будет производиться только среди них. Использование клавиш управления курсором может привести к стиранию набора в поле, клавиши “Home“ (“\“) и “End“ (“КОН“) также очищают поле и помещают курсор в корневой каталог или последний подкаталог. Вход в выбранный подкаталог производится нажатием клавиши “Enter“ (“ВВОД“). Путь из корневого каталога в выбранный подкаталог

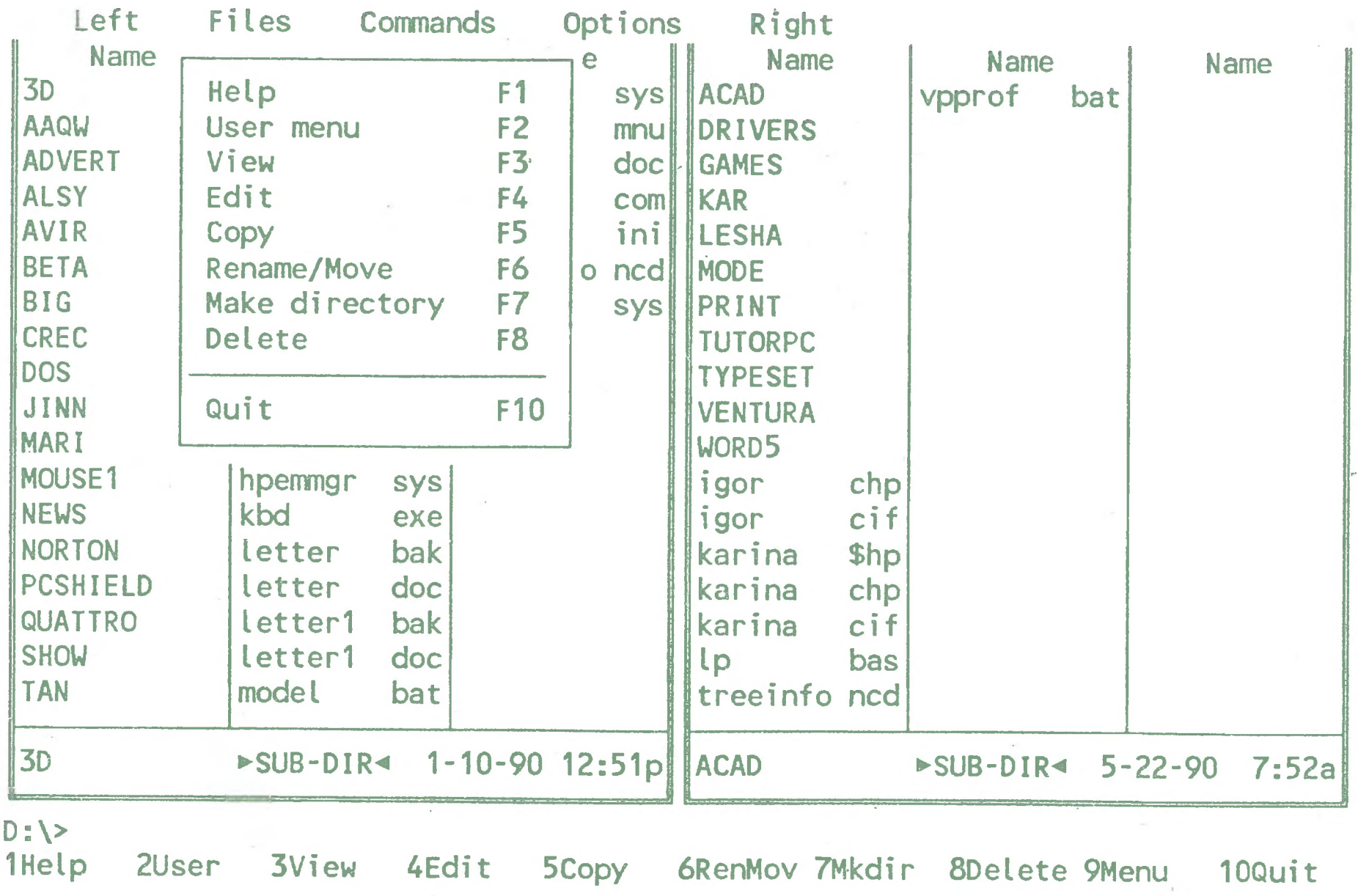


Рис.6

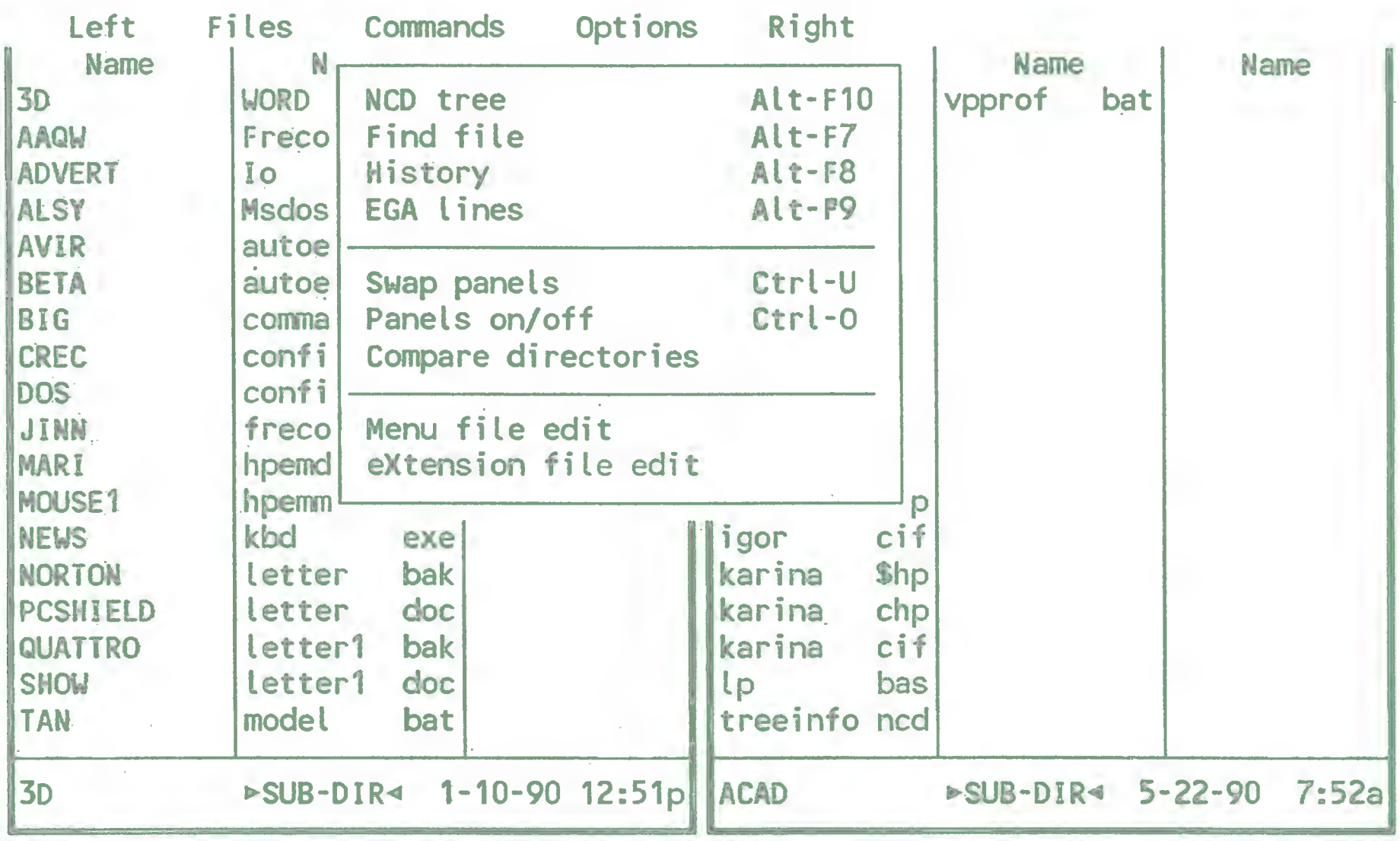


Рис.7

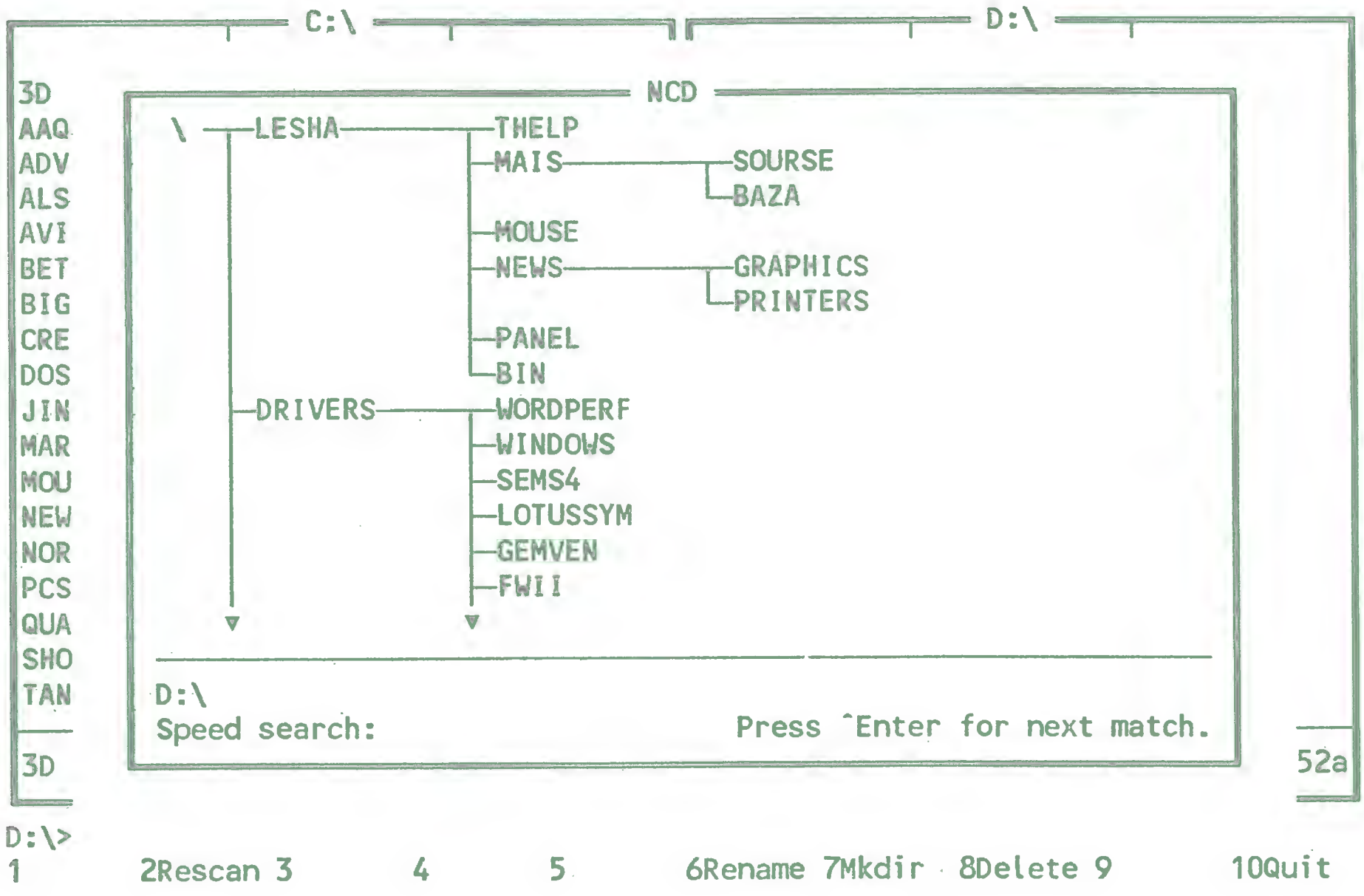


Рис.8

после этого будет отражен в верхней строчке панели. Помимо входа-выхода данная команда позволяет осуществлять бырый поиск другого директория, а также дает возможность создавать, удалять или переименовывать подкаталоги в текущем логическом дисководе. Производится это с помощью функциональных клавиш:

F2(Ф2) — просмотр содержимого дискеты (жесткого диска). NC строит файл “treeinfo.ncd“ для быстрого поиска и сохраняет в нем данные. Если какие-либо

программы внесли изменения в каталог, то, используя описываемую клавишу, необходимо повторно построить дерево NC текущего логического дисковода, что, в свою очередь, приведет к созданию свежего файла;

F6(Ф6) — переименование директория. После активизации наборного поля необходимо ввести новое мя или откорректировать старое. Отказ от выполнения процедуры производится с помощью клавиши "Esc"("КЛЮЧ");

F7(Ф7) — создание нового подкаталога в текущем директории;

F8(Ф8) — удаление подкаталога, помеченного курсором. Следует отметить, что этой операции поддается только пустой подкаталог.

2. Команда "Find File" осуществляет поиск заданного файла (группы файлов) по всем директориям активного дисковода, выводит на экран имена директориев, в которых обнаружены требуемые файлы, и предложит войти в один из них (рис.9). 5*) При необходимости можно произвести новый поиск ("New search") или выйти из команды поиска файлов ("Quit FF").

Как известно, имя файла в формате MS-DOS может иметь две части: собственно имя (длина не более 8 символов) и расширение (длина не более 3 символов). Использование специальных знаков "*" и "?" позволяет производить поиск файлов, имеющих в своих именах общие признаки. Например, имени файла в виде "A*" приведет к выбору файлов с именами "A", "AB", "ABC" и т.д. Символом "?" можно заменить одну букву. Например, в случае набора имени в виде "A?" будут выбраны файлы "AB", "AC", "AD" и т.д., а набор вида "????" позволит выбрать все файлы с именами длиной не более 5 символов. Аналогичным образом можно использовать вышеупомянутые знаки и в расширениях.

3. Команда "History" (рис.10) позволяет просмотреть последние 15 директив, выполнявшихся во время текущего сеанса NC и, выбрав одну, вновь выполнить ее. 6*) Директивы (команды) MS-DOS запоминаются по принципу FIFO ("first-in, first-out" — первым пришел, первым вышел), т.е. в порядке поступления.

4. Команда "EGA-lines" изменяет количество выводимых на экран строк (24 или 43 для EGA и 25 или 50 для VGA). Увеличение числа строк повышает информативность панелей NC. Необходимо предупредить пользователей, что в не-

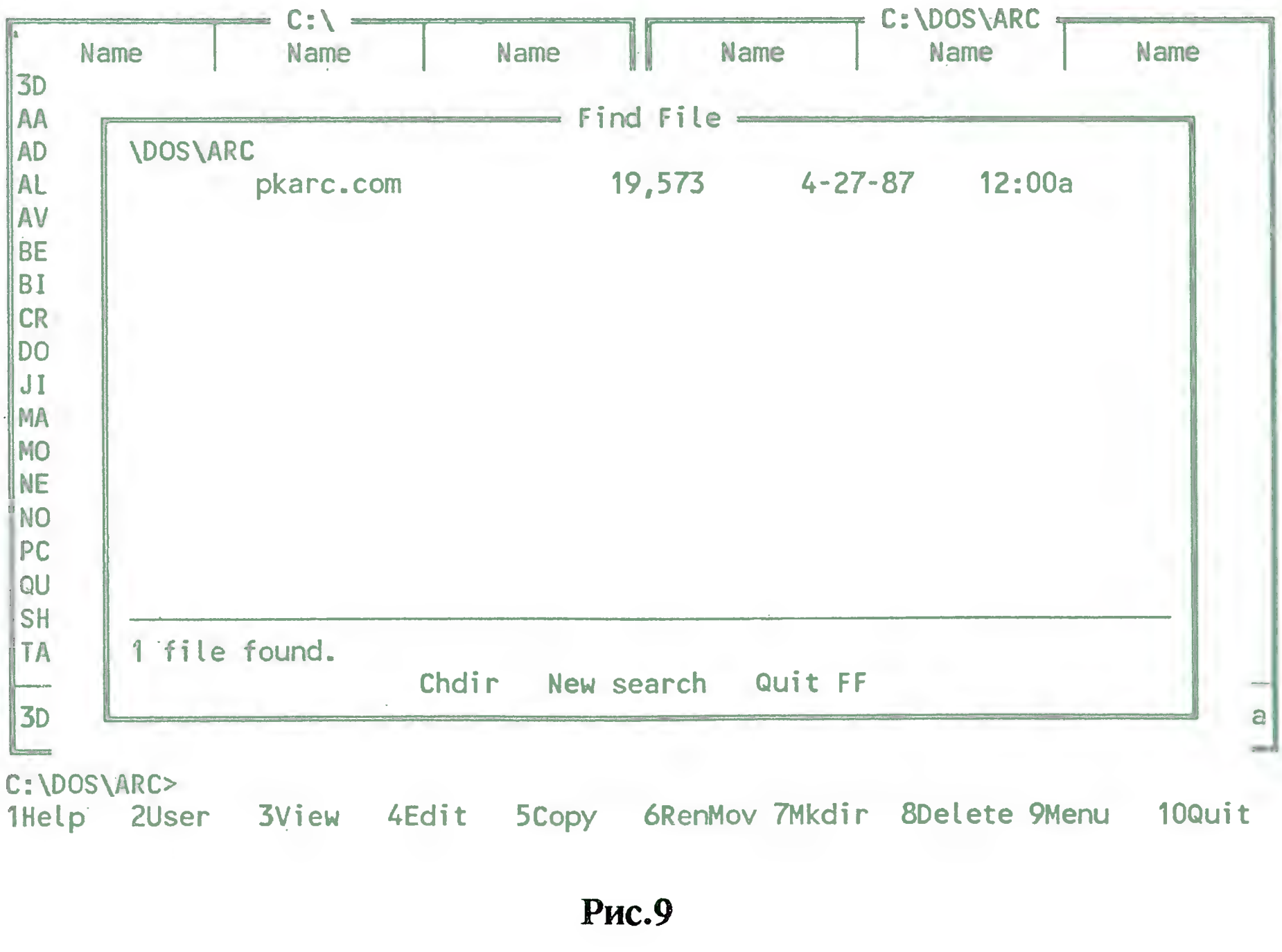


Рис.9

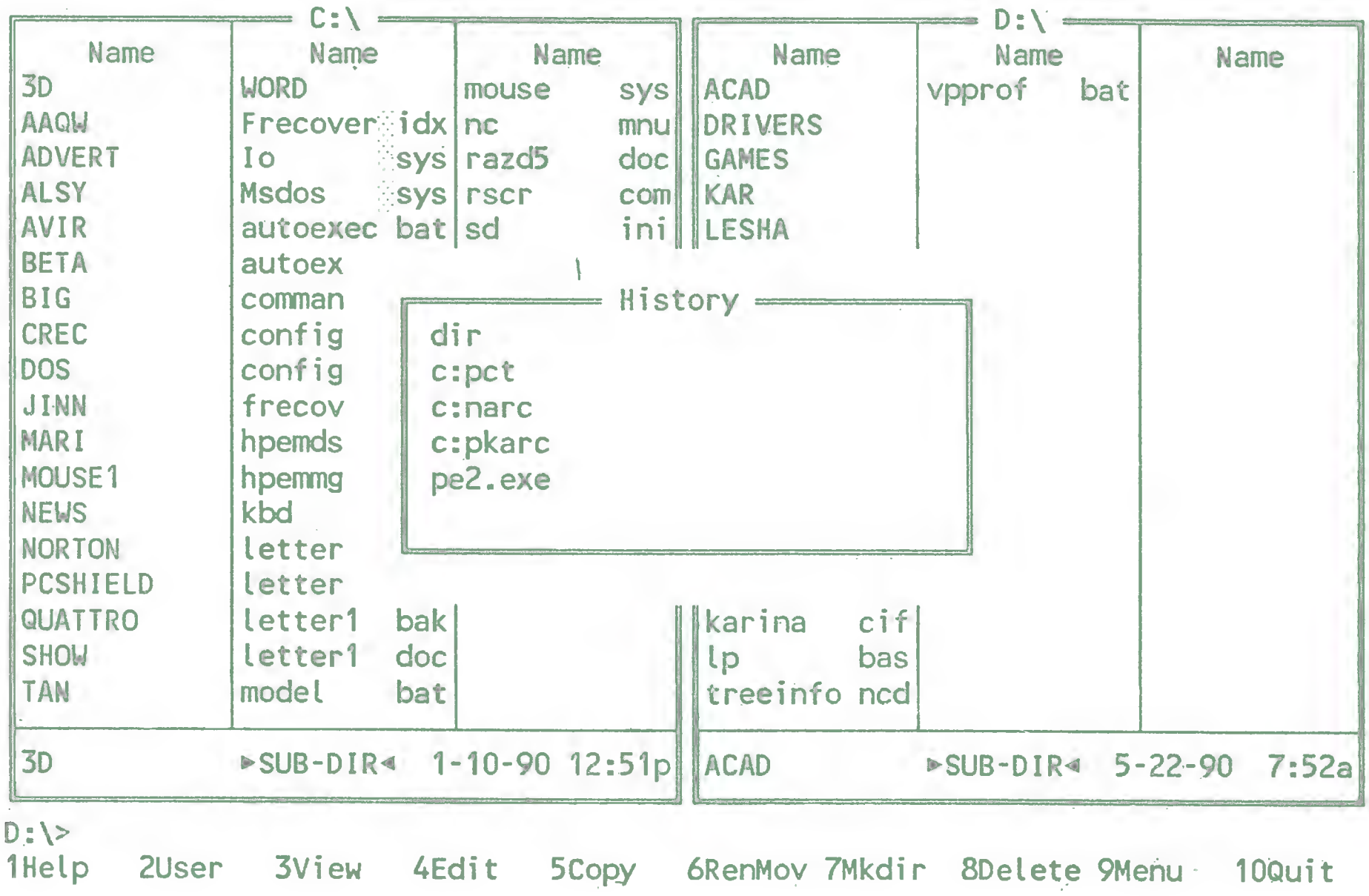


Рис.10

которых системах драйвер ANSI..SYS, загружаемый в конфигурационном файле, ограничивает число выводимых на экран строк первыми двадцатью пятью строками. Для устранения этого явления необходимо заменить файл ANSI.SYS. Очевидно, что команда “EGA-lines“ может выполняться только при наличии EGA- или VGA- адаптера. 7*)

5. Команда “Swap panels“ осуществляет перестановку панелей местами. 8*)

6. Команда “Panels on/off“ производит включение или выключение изображения панелей на экране монитора.9*)

7. Команда “Compare directories“ позволяет сравнить содержимое двух директориев. В случае идентичности будет выдана реплика “The two directories appear to be identical.“ При несовпадении каких-либо параметров файлы выделяются подсветкой. Следует отметить, что данная команда может быть полезна для проверки правильности копирования.

8. Команда “Menu file edit“ позволяет создавать, корректировать или удалять процедуры, используемые в меню пользователя (рис. 11). Основные правила оформления процедур таковы:

- первой строкой может быть строка комментариев, но при этом она должна начинаться с символа (апостроф). Наличие данной строки не обязательно;
- следующая строка является обязательной и должна состоять из двух частей: метки — символы и метки — слова. Метка — символ показывает, какую клавишу следует нажимать для выполнения пункта меню. В перечне процедур не должно быть двух одинаковых меток — символов. Метка — слово играет роль краткого наименования процедуры в меню пользователя и может представлять собой любое слово или набор символов длиной не более 60 байт;
- далее должна идти строка с одной из команд MS-DOS, которая будет выполняться;
- в следующих строках могут быть дополнительные команды MS-DOS для выполнения процедуры.

Необходимо учитывать, что данная команда NC позволяет работать с текста-

```
Edit: D:\nc.mnu                          *  Line 13    Col 8      30,387 Free   EOF
p: PCTools
   c:pct
   cls
e: EDITOR
   a:pe2
   cls
f: Format 730K
   c:
   f730 b:
   cls
n: NARC
   c:
   narc
```

```
                              User Menu Help
File format for user-defined menus:

' comment                    Comment line, ' must be in first column
m: Menu Label                Appears in the pop-up menu, with hot key 'm'
       first command         Any DOS command, must be indented
       command...            Any additional commands
```

```
1Help   2Save   3       4       5       6       7Search 8       9       10Quit
```

Рис.11

ми процедур не только для главного меню, но и для локальных меню, файлы которых находятся в активных в текущий момент директориях.

9. Команда “eXtension file edit“ позволяет создавать (корректировать) записи, определяющие реакцию NC на файлы, выбранные пользователем, в зависимости от расширения этих фалов. При выполнении этой команды на экран выйдет изображение, показанное на рис. 12. В нижней части кадра приведены правила создания записей:

```
Edit: C:\norton\nc.ext                   *  Line 11    Col 39     29,892 Free   110
prg: type !.!>prn
crg: type !.!>prn
bat: type !.!>prn
mnu: type !.!>prn
me : type !.!>prn
txt: type !.!>prn
doc: type !.!>prn
bak: type !.!>prn
:    type !
*:   rem no run file for this extension
```

```
                            Extension File Help
Format of the nc.ext file:          !.!     Filename with extension
txt: edit !.!                       !:      Drive letter (followed by ':')
 ▲    └──┘                          !\      Path name
 │       └── DOS command             !!      !
 └── File extension                  !       Filename without extension
```

```
1Help   2Save   3       4       5       6       7Search 8       9       10Quit
```

Рис.12

- в первых трех позициях первой строки должно быть указано расширение файла, на которое должна отреагировать система;
- после расширения должно идти двоеточие и пробел;
- затем идет команда MS-DOS. В команде MS-DOS должен быть указан один из мнемокодов:

!.! — для имени файла с расширением;
!: — для обозначения драйвера (следующего перед ´:´);
!\ — для имени пути;
!! — для восклицательного знака;
! — для имени файла без расширения.

Для выполнения нескольких команд подряд можно создать командный файл, который, в свою очередь, вызывается вышеуказанным способом.

Выбор управляющего слова “Option“ в верхней строке — подсказке приведет к активизации меню опций NC (рис. 13). Фиксация этих опций, как и фиксация вида сортировки файлов, отмечается знаком “ ± “. Опции NC представляют пользователю следующие услуги:

1. “Color...“ позволяет зафиксировать вид расцветки информации, выводимой на экран. После активизации меню пользователю будет предложен следующий выбор:

предложен следующий выбор:
“B&W — если компьютер оборудован монохромным дисплеем;
“Color“ — если компьютер оборудован цветным дисплеем;
“Lap-tor“ — если в распоряжении пользователя портативный компьютер.

Наличие данной опции, однако, не означает запрета на использование, к примеру, пункта “Color“ для монохромного дисплея. В этом случае изображение будет выводиться с градациями серого.

2. “Auto menus“ позволяет организовывать автоматический вывод меню пользователя.

3. “Path prompt“ дает возможность отображения пути, пройденного к текущему директорию, на командной строке.

Рис.13

4. “Key bar“ включает/выключает отображение нижней строки — подсказки.

5. “Full screen“ организует вывод информации на весь экран.

6. “Mini status“ включает в состав панелей строку с информацией о выбираемом или помеченном файле (группе помеченных файлов).

7. “Ins moves down“ позволяет включать/выключать перемещение курсора при пометке (отмене метки) файлов при использовании клавиши “Ins“ (“ВСТ“).

8. “cLock“ организует вывод показаний часов в правый верхний угол. Отображаемое время берется из таймера ЭВМ и может быть подвергнуто коррекции с помощью команды операционной системы.

9. “Editor...“ дает возможность переключать использование редактора, выбранного пользователем и внутреннего редактора NC. Производится это посредством выбора соответствующего управляющего слова:

“Built-in“ — встроенный,

“External“ — внешний.

Произведенный выбор будет впоследствии отмечен знаком “ ± “. При этом следует иметь в виду, что для использования внешнего редактора в качестве редактора NC его необходимо иметь в текущем каталоге. Возможность использования встроенного редактора при этом остается, для чего следует воспользоваться комбинацией клавиш “Alt-F4“ (“ДОП-Ф4“). Аналогичным обрезом можно использовать внешний редактор.

10. “Save setup“ запоминает настройку информационного табло для вывода в виде, установленном пользователем.10*)

IV. Операции копирования или перемещения могут быть выполнены для группы помеченных файлов. Пометка или отмена меток файлов может производиться по одному с помощью клавиши “Ins“ (“ВСТ“) или группой в одном каталоге с помощью серых клавиш “ + “ и “-“ ПЭВМ IBM и их клонов (“ + “ и “-“, находящихся в крайнем правом ряду для ПП ЭВМ серии ЕС). Если при копировании (перемещении) в выбранный директорий там уже есть файлы с такими же именами, то NC потребует сделать выбор:

- “Overwrite“, новый файл записывается вместо обнаруженного, при совпадении имен нескольких файлов на каждый делается индивидуальный запрос;

- “All“, групповая запись без индивидуального запроса на каждый файл.

Ранее записанные файлы при этом теряются;

- “Skip“, очередной помеченный для копирования (перемещения) файл пропускается, метка с него снимается.

V. Текст или иная информация (например, результат решения задачи), занесенная в файл “DIRINFO“, будет отображена в нижней части панели “INFO“. Этот файл может быть полезен для напоминания о назначении программ текущего каталога, типовых приемов работы, какой-либо информации о текущем директории, необходимой пользователю. Для создания файла “DIRINFO“ необходимо инициировать вышеупомянутый режим, а затем воспользоваться помощью функциональной клавиши F4 (Ф4).

VI. Движение по каталогу файлов производится с помощью клавиш управления курсором, при этом курсор помечает файл, на котором в данный момент находится система.

“↓“ — перемещает курсор в каталоге на имя следующего файла;

“↑“ — перемещает курсор в каталоге на имя предыдущего файла;

“→“ и “←“ — эти клавиши работают в режимах “Brief“ и “Tree“ и перемещают курсор в соседний столбец соответственно. Если соседний столбец пуст, то курсор перемещается на имя последнего файла в каталоге. В режиме “Full“ эти клавиши управляют курсором в командной строке:

“Home“ (“↖“) — перемещает курсор в начало каталога;

“End“(“КОН“) — перемещает курсор в конец каталога активной панели;

"PgDn"("⏏") — выводит на активную панель следующую страницу каталога;

"PgUp"("⏏") — возвращает на активную панель предыдущую страницу каталога.

VII. Создание подкаталога возможно с помощью функциональной клавиши F7(Ф7). Эта операция позволяет организовать подкаталог на активной панели файлов или дерева. Название подкаталога может состоять не более чем из двух частей: основного имени длиной до восьми символов и расширения длиной до трех символов. Вход в подкаталог осуществляется выбором его клавишами управления курсором с последующим нажатием клавиши "Enter" ("ВВОД"). Для выхода из подкаталога необходимо перейти в его первую строку и нажать клавишу "Enter" ("ВВОД"). Первая строка подкаталога имеет в поле "Name" точку (-и) вместо имени. Перед сменой дискеты желательно выйти из подкаталога, иначе в дальнейшем могут возникнуть затруднения.

VIII. При наличии в тексте команды (опции) выделенной буквы для выполнения такой команды (опции) достаточно нажать клавишу с выделенной буквой.

IX. Поиск имени файла в оглавлении на активной панели можно производить с помощью клавиши "Alt"("ДОП"). Для этого необходимо ее нажать и, удерживая, набрать первый символ имени требуемого файла (для дальнейшего набора клавишу "Alt" удерживать не обязательно). Курсором будет выделен первый встреченный файл, набор для которого совпадает с началом его имени. Если таких файлов будет несколько, то их перебор можно осуществить с помощью комбинации клавиш "Ctrl-Enter"("УПР-ВВОД").

X. Краткий перечень действий NC с использованием функциональных клавиш, клавиш "Ctrl"("УПР"), "Alt"("ДОП") и символьных приведен в таблице.

Примечания

1*) NC еще иногда называют программной оболочкой, так как она находится резидентно, т.е. постоянно, и при этом как бы обволакивает работу MS-DOS и активной программы.

2*) Если на экране ничего нет, это еще не значит, что NC не активен. Опытный пользователь может без особого труда убрать или восстановить отображаемые части NC на экране. Производится это одновременным нажатием клавиш "Ctrl-O" ("УПР-О") для обоих панелей, "Ctrl-F!" ("УПР-Ф1) — для левой панели, "Ctrl-F2" ("УПР-Ф2) — для правой панели, "Ctrl-B" ("УПР-В") — для нижней строки — подсказки.

3*) Повторное чтение каталога логического дисковода либо выбор другого дисковода с чтением соответствующего каталога можно осуществить с помощью одновременного нажатия клавиш "Alt-F1 ("ДОП-Ф1") или "Alt-F2" ("ДОП-Ф2"). Если в выбранном дисководе отсутствует дискета или запись сделана в другом формате, то NC предложит выбрать другой дисковод или прекратить работу.

4*) Выполнение команды "NCD — tree" можно произвести одновременным нажатием клавиши "Alt-F10"("ДОП-Ф10").

5*) Выполнение команды "Find file" можно произвести одновременным нажатием клавиш "Alt-F7" ("ДОП-Ф7").

6*) Выполнение команды "History" можно произвести одновременным нажатием клавиш "Alt-F8" ("ДОП-Ф8").

7*) Выполнение команды "EGA — lines" можно произвести одновременным нажатием клавиш "Alt-F9" ("ДОП-Ф9").

8*) Выполнение команды "Swap panels" можно произвести одновременным нажатием клавиш "Ctrl-U" ("УПР-U").

9*) Выполнение команды "Panels on/off" можно произвести одновременным нажатием "Ctrl-O" ("УПР-О").

10*) Выполнение операции "Save setup" можно произвести одновременным нажатием клавиш "Shift-F9" ("|-Ф9").

11*) Вход в подкаталог можно произвести одновременным нажатием клавиш "Ctrl-PgDn" (предварительно пометив подкаталог курсором), а выход — одновременным нажатием "Ctrl-PgUp".

****

В СССР получили распространение несколько версий NC. Данная статья охватывает опыт работы с версией 2.0, но пользователь, освоивший описанные действия, сможет выбрать для себя наиболее подходящую версию.

Таблица

		"Ctrl" ("УПР")	"Alt" ("ДОП")	"Shift" ("↑")
F1	Подсказка	Вкл/Откл левой панели	Изменение дисковода левой панели	-
F2	Меню пользователя	Вкл/Откл правой панели	Изменение дисковода правой панели	-
F3	Просмотр	-	Просмотр файла	Просмотр заданного файла
F4	Коррекция файла	-	Активизация редактора	Коррекция заданного файла
F5	Копирование	-	-	Копирование заданного файла
F6	Перемещение	-	-	Перемещение заданного файла
F7	Создание подкаталога	-	Поиск файла	Создание подкаталога
F8	Удаление	-	История команд	Удаление заданного файла
F9	Меню NC	-	Количество строк	Запоминание настройки
F10	Завершение работы NC	-	Дерево подкаталогов	-
\	-	Возврат в корневой каталог	-	-
B	-	Вкл/откл нижней линейки меню	-	-
E	-	Предыдущая команда	-	-
I	-	Изменение активной панели	-	-
J	-	Имя файла в командную строку	-	-
L	-	Переключение вида информации	-	-
M	-	Редактор	-	-
N	-	Включение меню NC	-	-
O	-	Вкл/Откл панелей	-	-
P	-	Вкл/откл неактивной панели	-	-
R	-	Повторное чтение активного каталога	-	-
U	-	Перестановка панелей местами	-	-
X	-	Следующая команда	-	-

ГЛОССАРИЙ

A

All — все, вс.

Alt (сокр. от alter) — "альтернативная" клавиша, изменяющая код клавиш, нажимаемых вместе с ней (для ПП ЭВМ ЕС1840 и 1841 это клавиша "ДОП").

Appear — казаться.

Auto — автоматический, автоматическое.

B

BS (сокр. от backspace) — клавиша, вызывающая возврат на один символ с удалением предыдущего.

B&W (сокр. от black and white) — черно-белый.

Bar — линейка, строка.

To be — 1) быть, существовать, находиться; 2) служит для образования глагольных форм; 3) выражает долженствование, возможность, намерение.

Brief — краткий.

Built-in — встроенный.

C

Cd (сокр. от change directory) — изменить каталог, перейти в другой каталог.

Clock — часы.

Char (сокр. от character) — символ, знак литера.

Color — цвет.

Command — команда.

Commander — командир, начальник, в данном случае программа, отдающая (выполняющая) команды.

Compare — сравнивать.

Copy — 1) экземпляр, копия; 2) копировать.

Ctrl (сокр. от control) — регистровая клавиша, используемая для ввода управляющих кодов.

Cursor — курсор; метка на экране, указывающая место, где производятся действия.

D

Date — дата, число.

Del (сокр. от delete) — 1) исключать, стирать, удалять; 2) клавиша, используемая для удаления символа, помеченного курсором (в ПП ЭВМ типа ЕС1840 или 1841 это клавиша "УДЛ").

Dirinfo (сокр. от directory information) — информация о директории.

Down — вниз.

E

Edit — редактировать.

EGA (сокр. от Enhanced Graphics Adapter) — усовершенствованный графический адаптер.

End — 1) конец; 2) служебное слово, указывающее на конец программной конструкции; 3) клавиша, используемая для задания перемещения курсора программе (в ПП ЭВМ типа ЕС1840 или 1841 это клавиша "КОН").

Enter — 1) ввод; 2) служебное слово, указывающее на ввод программной конструкции; 3) управляющая клавиша, служащая для ввода набранной информации (в ПП ЭВМ типа ЕС1840 или 1841 это "Ввод").

Esc (сокр. от escape) — 1) выход; 2) завершение выполнения блока программы или работы в некотором режиме, предусмотренном программой; 3) клавиша, используемая для прерывания и завершения работы программы (если предусмотрено программой).

Extention — расширение.

External — внешний.

F

File — файл; последовательность или множество однотипных записей; поименованная целостная совокупность данных на внешнем носителе.

Find — 1) поиск; 2) находить, обнаруживать.

Full — полный.

H

Help — подсказка, диалоговая документация; средство интерактивной системы, позволяющее пользователю получить информацию об операциях и командах, допустимых в текущем состоянии системы.

History — история.

Home — 1) начало экрана, левый верхний угол экрана дисплея; 2) клавиша, используемая для перевода курсора в начало экрана (строки).

I

Identical — идентичный.

Info (сокр. от information) — информация.

INS (сокр. от insert) — 1) вставка; 2) вставлять; 3) клавиша позволяющая реализовать режим вставки символов в месте, указанном курсором.

K

Key — клавиша, ключ.

L

Laptop — портативная ЭВМ.

Left — левый, налево, слева.

Line — линия, строка.

M

Make — делать, производить, совершать.

Menu — меню; изображаемый на экране дисплея список команд или вариантов ответа, из которого пользователь выбирает необходимый вариант, вводя номер (букву) или указывая на пункт меню курсором.

Menu bar — линейка меню; строка окна или экрана дисплея, в которой перечислены пункты меню.

Ministatus — мини-статус; краткое сообщение об основных характеристиках файла или группы файлов.

Mkdir (сокр. от make the directory) — сделать (организовать, образовать каталог (подкаталог).

Move — пересылать, перемещать данные из одной области памяти в другую.

Movement — пересылка, перемещение.

MS-DOS — операционная система для 16-разрядных персональных ЭВМ на базе микропроцессоров, совместимых с Intel 8086.

N

Name — имя, название.

NC — сокращенно от Norton Commander

NCD-tree (сокр. от Norton Commander directory tree) — "дерево" каталога в системе NC.

New — новый.

Next — следующий.

On/off — включить/выключить.

Option — опция, вариант, средство.

Other — другой, другое, иной, иное.

Owerwrite — затирать, записывать данные в область носителя, занятую другими данными.

P

Panel — панель, табло.

Path — 1) путь доступа, маршрут; 2) список каталогов, в которых следует искать файл.

PsDn (сщкр. от Page Down) — клавиша, используемая для задания переме-

щения курсора на страницу вниз.

PgUp (сокр. от Page Up) — клавиша, используемая для задания перемещения курсора на страницу вверх.

Previous — предыдущий, предшествующий.

Prompt — быстрый, немедленный.

Q

Quit — оставлять, покидать, прекращать, выбывать.

R

Rename — переименовать.

Ren/Mov (сокр. от rename/move) — переименовать/переместить.

Re-read — перечитать, прочитать повторно.

Right — правый.

S

Save — сохранять, записывать.

Save setup — сохранять настройку, записывать настройку.

Screen — экран.

Search — 1) поиск, перебор; 2) искать.

Search area — область поиска.

Select — выбирать, выделять, помечать.

Setup — настройка, установка.

Shift — 1) сдвиг; 2) смена регистра; 3) управляющая клавиша для изменения соответствия клавиш клавиатуры и выдаваемых ими кодов или способа интерпретации кодов.

Size — размер, длина.

Skip — 1) пропуск; 2) пропускать, игнорировать.

Speed — 1) скорость, быстрота; 2) скоростной, быстрый.

Status — состояние.

Swap — переставлять, менять местами.

Switch — 1) переключатель; 2) переключать.

T

Tab (сокр. от tabulation) — 1) символ табуляции, табуляция; 2) клавиша клавиатуры терминала, выдающая символ табуляции и вызывающая перемещение курсора к следующей позиции табуляции.

Time — время.

Toggle — 1) переключатель, ключ; 2) изменять состояние на противоположное.

Tree — 1) дерево; 2) конечное множество элементов, в котором выделен один элемент (корень), а остальные элементы разбиты на непересекающиеся множества (поддеревья), каждое из которых является деревом; 3) структура данных, представляющая дерево.

U

Under — в соответствии, согласно.

Unselect — отменить выборку, выделение, пометку.

Unsorted — несортированный.

User — пользователь, потребитель.

V

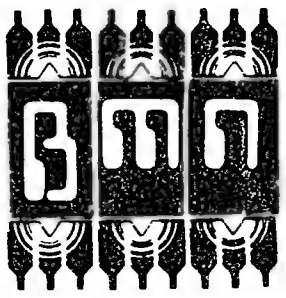
VGA (сокр. от Video Graphic Adapter) — видеографический адаптер.

View — вид, поле зрения, взгляд, мнение, представление, видимое изображение.

W

Word — слово.

Автор благодарит своих коллег А.Смурова, С.Долгирева, А.Кирилова и В.Майорова за помощь и консультации, оказанные ему в процессе подготовки статьи.



Вы пишете на языке С и хотите, чтобы ваши программы имели столь же красивый и удобный вид, как всем известные Norton Commander, dBase или Multi-Edit? Если да, то имеет смысл воспользоваться пакетом Vitamin C. Эта статья позволит вам его освоить.

В.В.Слободчук

ПАКЕТ "VITAMIN C"

ПОДГОТОВКА VITAMIN C К РАБОТЕ

Если вы пользуетесь MicroSoft C, то:

1) Скопируйте 8 файлов с расширением .H в директорию, где находятся все .H-файлы вашего компилятора.

2) Найдите в файле VCSTDIO.H строку

```
#define ???
```

и преобразуйте ее в

```
#define MSC.
```

3) Найдите в файле VCSTDIO.H строку

```
#include <mscstdio.h>
```

и преобразуйте ее в

```
#include <stdio.h>.
```

4) Скопируйте файлы с расширением .LIB в директорию, где находятся все библиотечные файлы вашего компилятора.

Vitamin C готов к использованию.

Если вы применяете какой-либо другой C-компилятор (Turbo C, Lattice C и т.д.), в п.п. 2 и 3 вместо сокращения "MSC" нужно употреблять сокращения TC, LC и т.д. (см. комментарий в начале файла VCSTDIO.H).

Кроме того, Вам не подходят уже готовые библиотеки Vitamin C (они собраны для MicroSoft C). Необходимо перекомпилировать все исходные тексты (все файлы с расширением .C) и собрать новую библиотеку. В эту библиотеку следует включить также модули VCVIOxxy.OBJ и VCVGAxxy.OBJ, где вместо xx должно быть TC или LC, или AZ и т.д. y — обозначение модели памяти (l — large, s — small,...).

НЕМНОГО ЖАРГОНА

Вместо слов "пакет Vitamin C" далее будем употреблять сокращение VC.

АТРИБУТЫ — это просто цвет символов и фона. Например, можно сказать, что цвет атрибутов строки — это ярко-красные буквы на синем фоне.

ЭКРАННЫЙ ВЫВОД — набор функций, выводящих что-либо на экран.

ЭКРАННЫЙ ВВОД — набор функций, вводящих данные с клавиатуры(!)

ОСНОВНАЯ КОНЦЕПЦИЯ

В сущности, VC — это библиотека самых разнообразных функций (всего их более 300), позволяющих создавать окна, меню, управлять курсором и т.д.

Чтобы пользоваться ими, от вас требуется выполнение одного условия: НЕ УПОТРЕБЛЯТЬ никаких функций экранного ввода/вывода, кроме функций VC.

```
puts( )    <- нет !        vcputs( ) , <- да !
printf( )  <- нет !        vcgets( )   <- да !
scanf( )   <- нет !        vcprint( )  <- да !
```

Структура всех программ, использующих VC, однотипна:

```
#include <vcstdio.h>
        .
        .
        .
main( )
{       .
        .
        .
    vcstart(параметр);
        .
        .
        .
    vcend(параметр);
        .
        .
        .
}
```

ПЕРВОЙ СТРОКОЙ ТЕКСТА, отличной от комментария, должна быть инструкция включения файла VCSTDIO.H. ПЕРВОЙ ИСПОЛНЯЕМОЙ СТРОКОЙ программы должен быть вызов функции vcstart(). Перед ЗАВЕРШЕНИЕМ программы должна быть вызвана функция vcend(). Остальное не регламентируется.

Функция vcstart( ): `VOID vcstart(option)`
`COUNT option;`

Эта функция инициализирует VC. Ее параметр может принимать значения CLRSCRN, тогда при инициализации VC экран очищается; либо SAVESCRN — тогда экран остается без изменений.

Не удивляйтесь, увидев в описаниях незнакомые типы COUNT, TEXT и др. или константы наподобие SAVESCRN. Эти стандартные для VC типы и константы определены в файле VCDEF.H. Он автоматически подключается к вашей программе.

Функция vcend( ): `VOID vcend(option)`
`COUNT option;`

Параметр option может иметь значение CLOSE, тогда все открытые на момент завершения программы окна закрываются; либо NOCLOSE — тогда результаты программы остаются на экране.

ПРОСТЕЙШИЕ ФУНКЦИИ

Чтобы приостанавливать работу приводимых далее примеров в нужном месте, нам потребуется функция getone( ). Эта функция — аналог

Функция getone( ):	COUNT getone( )

стандартной функции getch( ), хотя getone( ) дополнительно выполняет большой набор важных действий. getone( ) ждет нажатия любой клавиши, после чего возвращает код этой клавиши, и программа продолжит выполнение.

Парной функцией является ungetone( ). Это полный аналог стандартной функции ungetch( ).

Функция ungetone( ):	COUNT getone(aus) COUNT aus; код возвращаемой клавиши

Пример:

```
.
.
.
getone( );          /* Требуется нажатие клавиши */
.
.
.
ungetone('B');
getone( );          /* Не требуется нажатия клавиши.
.                      getone( ) "считает", что была
.                      нажата клавиша <B> */
.
```

Для перемещения курсора в любую точку экрана используется функция at( ).

Функция at( ):	VOID at(row,col) COUNT row, col;	новые координаты курсора, считая от (0,0)

Выполнение инструкции at(10,13) поместит курсор в 10-ю строку и 13-й столбец экрана. Печать строки символов производит функция say( ).

Функция say( ):	VOID say(str) TEXT *str; выводимая строка

Функция atsay( ) — комбинация двух предыдущих функций.

Функция atsay( ):	VOID atsay(row,col,str) COUNT row, col; TEXT *str;	координаты начала строки выводимая строка

Пример:

```
#include <vcstdio.h>
main( )
{
```

```
COUNT y,x;
TEXT  line[30];
   vcstart(CLRSCRN);
   y = 12; x = 7;
   at(y,x);
   say("Первая строка");
   getone( );        /* Пауза — полюбуйтесь на строку */
   strcpy(line,"Вторая строка");
   atsay(y,30,line);
   getone( );        /* Пауза — полюбуйтесь на ваши строки */
   vcend(NOCLOSE);
}
```

АТРИБУТЫ

VC содержит 8 глобальных переменных для задания цвета, которые всегда доступны в ваших программах. Их имена говорят сами за себя:

vc.black	vc.cyan
vc.blue	vc.brown
vc.green	vc.red
vc.magenta	vc.white

Кроме того, имеются:

- переменная для задания яркости — vc.bold;
- переменная для задания мерцания — vc.blink;
- переменная для задания фона — vc.bg.

Если вы хотите воспользоваться умолчательными атрибутами, используйте переменную vc.dflt.

Пример: vc.brown + (vc.bg*vc.green) — коричневый текст на зеленом фоне.

vc.brown + vc.bold + (vc.bg*vc.green) — желтый текст на зеленом фоне.

vc.brown + vc.bold + vc.blink + (vc.bg*vc.green) — мерцающий желтый текст на зеленом фоне.

vc.dflt — кто-то раньше (Вы или VC) задал атрибуты, используются эти атрибуты.

Более сложный вариант atsay( ) — функция xatsay( ):

Функция xatsay( ):

```
VOID xatsay(row,col,str,attr)
COUNT  row, col;   координаты начала строки
TEXT   *str;       текст сообщения
COUNT  attr;       атрибуты сообщения
```

Для очистки экрана можно использовать функцию erase( ). Эта

Функция erase( ):

```
VOID erase( )
```

функция очищает экран с цветом фона по умолчанию и помещает курсор в точку с координатами (0,0).

Более сложный вариант — функция xerase( ). Эта функция очищает экран с указанным цветом фона.

```
Функция      VOID xerase(attr)
xerase( ):   COUNT attr;    атрибуты фона
```

Пример:
```
#include <vcstdio.h>
main( )
{
COUNT color;

   vcstart(CLRSCRN);
   xatsay(10,13,"Пишем синим по красному",
                  vc.blue + (vc.bg*vc.red));
   getone( );              /* пауза — нажмите любую клавишу */
   erase( );               /* очистить экран; цвет фона — на усмотрение VC */
   color = vc.red + vc.bold + (vc.bg*vc.blue);
   xatsay(1,4,"Пишем ярко-красным по синему",color);
   getone( );              /* пауза — нажмите любую клавишу */
   xerase(color);          /* очистить экран; цвет фона — синий */
   getone( );              /* пауза — нажмите любую клавишу */
   vcend(NOCLOSE);         /* конец работы */
}
```

ЭКРАННЫЙ ВВОД

Аналогом стандартной функции gets( ) можно считать пару функций atget( )-readgets( ).

```
Функция      s = atget(row,col,fld,pic)
atget( ):    COUNT  s;          0 — если нет ошибок, иначе -1
             COUNT  row,col;    координаты для ввода
             TEXT   *fld;       указатель поля для ввода
             TEXT   *pic;       шаблон для ввода
Функция      COUNT readgets( )
readgets( ):        Возвращает код клавиши, вызвавшей завершение
                    ввода данных. Это могут быть:
             RET    — клавиша <RETURN> (<ENTER>)
             ESC    — клавиша <ESC>
             QUIT_INPUT — обычно аналогично <ESC>
```

atget( ) не производит ввода, а только определяет, где и что будет вводиться (определяет так называемое ПОЛЕ ВВОДА). Ее исполнение невидимо на экране. Собственно ввод производит функция readgets( ). Координаты row,col определяют, где будет выведено эхо вводимых символов.

Пример:
```
#include <vcstdio.h>
main( )
{
TEXT born[5];
  vcstart(CLRSCRN);          /* инициализация VC */
  empty(born,5);             /* заполнить born пробелами */
  atsay(10,15,"Введите год рождения: ");
  atget(10,30,born,"9999"); /* в точке (10,30) ждем
                               ввода 4 цифр */
```

```
    readgets( );                    /* произвести ввод */
        /* нажмите клавиши <1>,<9>,<3>,<4> */
    atsay(20,1,"Вы ввели ");
    say(born);                      /* born = ="1934" */
    getone( );                      /* пауза — нажмите любую клавишу */
    vcend(NOCLOSE);
}
```

В этом примере вы встретились с одной из вспомогательных функций VC — empty( ). Она просто заносит указанное число пробелов в строку.

Функция empty( ):	VOID empty(str,len)		
	TEXT	*str;	строка, которую заполняют пробелами
	COUNT	len;	длина строки, включая NULL

Шаблон показывает, какой символ ожидается на соответствующем месте при вводе. Если вводимый символ не удовлетворяет шаблону, то этот символ игнорируется.

Шаблон может содержать такие символы:

x — в данной позиции допустим ввод любого символа;

X — то же самое, но если вводится буква, то она автоматически преобразуется в верхний регистр;

a — в данной позиции допустим ввод только буквы;

A — то же самое, но буква автоматически преобразуется в верхний регистр;

9 — в данной позиции можно вводить только цифру.

Примеры шаблонов: "aaaaa" — ожидается ввод пяти букв;
"99A" — ожидается ввод двух цифр и одной буквы. Буква будет преобразована в верхний регистр.

Шаблон может также иметь вид "#", "##", "###" и т.д. Такой шаблон обеспечивает ввод числа, причем эхо-изображение вводимых цифр будет формироваться справа налево, как в калькуляторе. Символы '#' нельзя употреблять с другими символами. Если вы зададите шаблон "9999a", то при вводе с помощью readgets( ) VC выдаст на экран пустое поле ввода длиной 5 символов. В это поле будет выводиться эхо вводимых с клавиатуры символов.

Кроме того, VC позволяет для удобства помещать в шаблон любые другие символы, например — '\', '(', '-' или даже буквы. Эти символы выводятся на экран на месте поля ввода при исполнении readgets( ). Если вы зададите шаблон "код 99-99(a)", то при вводе с помощью readgets( ) VC выдает на экран не пустое поле ввода, а такое: "код — ( )". Это более наглядно, чем пустое поле. В начальный момент при вводе курсор будет находиться после символов "код ". После ввода двух цифр курсор автоматически перепрыгнет через знак '-'. После ввода третьей и четвертой цифр курсор сам перепрыгнет через открывающую скобку и позволит ввести любую букву.

Пример:
```
#include <vcstdio.h>
main( )
{
TEXT *phone[10];
  vcstart(CLRSCR);          /* инициализация VC */
  empty(phone,10);          /* заполнить phone пробелами */
  atsay(10,15,"Номер телефона: ")
  atget(10,30,phone,
```

```
                    "999-99-99"); /* объявить ввод 7 цифр */
    readgets( );                  /* нажмите <1>,<2>,<3>...<7> */
                                  /* строка phone теперь
                                  содержит "123-45-67" */
    vcend(NOCLOSE);
}
```

Если при вводе по readgets( ) вы допустили ошибку, ошибочные символы можно исправить. Для этого используются стандартные клавиши управления курсором <СТРЕЛКА_ВПРАВО>, <СТРЕЛКА_ВЛЕВО>, <BackSpace>, а также <INS>, <DEL> и <ESC>.

Можно объявить несколько полей ввода несколькими функциями типа atget( ), а затем ввести ВСЕ данные с помощью ОДНОГО вызова функции readgets( ).

Пример:

```
#include <vcstdio.h>
main( )
{
TEXT string1[5], string2[10], string3[10];
TEXT picture[10];
  vcstart(CLRSCRN);          /* инициализация VC */
  empty(string1,5);          /* Очистить строки */
  empty(string2,10);
  strcpy(string3,"Vitamon C");
  atsay(1,2,"Введите что угодно: ");
  get(string1,"xxxx");

  strcpy(picture,"aAaAaAaAa");
  atsay(3,2,"Введите только буквы: ");
  get(string2,picture);

  atsayget(5,2,"Исправьте ошибку: ",string3,"aaaaaaaaa");

  readgets( );               /* ввести данные во все три поля
                             одним вызовом readgets( ) */
  vcend(NOCLOSE);
}
```

В этом примере вы встретились еще с двумя вариантами функции atget( ). Первый — get( ) — более простой. Он объявляет поле ввода (как и atget( )), но не в любом месте, а там, где на момент вызова get( ) оказался курсор. Второй — atsayget( ) — более сложный. Это комбинация функций atsay( ) и get( ).

Функция get( ):	s = get(fld,pic)		
	COUNT	s;	0 — если нет ошибок, иначе -1
	TEXT	*fld;	указатель поля для ввода
	TEXT	*pic;	шаблон для ввода
Функция atsayget( ):	s = atsayget(row,col,sfld,gfld,pic)		
	COUNT	s;	0 — если нет ошибок, иначе -1
	COUNT	row, col;	координаты курсора для вывода подсказки
	TEXT	*sfld;	текст строки подсказки

```
TEXT    *gfld;      указатель поля для ввода
TEXT    *pic;       шаблон для ввода
```

Совсем сложный вариант объявления поля ввода — функция xatget( ).

```
Функция     s = xatget(row,col,fld,pic,val,msg,help,act,nact)
xatget( ):  COUNT  s;          0 — если нет ошибок, иначе -1
            COUNT  row,col;    координаты для ввода
            TEXT   *fld;       указатель поля для ввода
            TEXT   *pic;       шаблон для ввода
            PFI    val;        указатель на проверяющую функцию
            TEXT   *msg;       указатель на статус-сообщение
            TEXT   *help;      указатель на ключ HELP-системы
            COUNT  act;        атрибуты поля в активном состоянии
            COUNT  nact;       атрибуты поля в неактивном состоянии
```

Весьма громоздкая функция! Но зато она многое умеет. Назначение многих параметров сейчас непонятно. Поясним некоторые.

val — указатель на функцию, проверяющую корректность введенных данных. Она вызывается после завершения ввода в объявленное поле. В качестве параметра такой функции передается указатель на поле ввода. Если его содержимое удовлетворительно, то функция проверки должна вернуть значение GOOD (равное 0), если же что-то не в порядке — вернуть значение BAD (ненулевое). readgets( ) проверяет это значение и будет заставлять вас вводить до тех пор, пока функция проверки не удовлетворится введенными данными. msg — текст, который появится в статус-окне при вводе в объявленное поле. Пока вы не знаете, что такое статус-окно, указывайте на этом месте NULL. То же относится к help.

Пример:

```
#include <vcstdio.h>
main( )
{
TEXT   fname[13];
TEXT   disk[2] = {'C','\000'};

   vcstart(CLRSCRN);                    /* инициализация VC */

   empty(fname,13);                     /* очисчтить fname */
   atsayget(10,2,"Введите имя файла: ",fname,"xxxxxxxx.xxx")
   atsay(12,2,"Введите имя диска (A-D): ");
   xatget(12,29,disk,"A",test,NULL,NULL,
               vc.black + (vc.bg*vc.white),
               vc.white + (vc.bg*vc.black));
   readgets();
   vcend(NOCLOSE);
}

COUNT test(TEXT *drvnum)
/* Проверяет содержимое *drvnum.
    Выход: 0 — если *drvnum указывает на строки
                  "A", "B", "C", "D";
           !0 — в противном случае.   */
{
```

```
        if (!strcmp(drvnum,"A")
         || !strcmp(drvnum,"B")
         || !strcmp(drvnum,"C")
         || !strcmp(drvnum,"D"))
          return(GOOD);                    /* Нормальное имя диска */
        else
          { bell( );                       /* Плохое имя — подать */
            return(BAD); }                 /* звуковой сигнал  */
}
```

Функция bell( ): VOID bell( )

ОБЛАСТИ SAY, AGET, NAGET

Все до сих пор использованные для ввода/вывода участки экрана можно условно разделить на три основные группы:

1. Участки, заполненные с помощью функций say( ), atsay( ), xatsay( ), erase( ), xerase( ) и т.д. Такие участки в VC называются областью SAY.

2. Участки, объявленные для ввода функциями get( ), atget( ), xatget( ) и т.д., но лишь те из них, в которые данные еще не вводятся (или уже были раньше введены). Эти участки "неактивного get" в VC называются областью NAGET.

3. Участок, объявленный для ввода функциями get( ), atget( ), xatget( ) и т.д., в который в данный момент вы вводите данные. Этот участок "активного get" в VC называется областью AGET. Для функции atsayget( ) (см. выше) строка подсказки относится к области SAY, а поле ввода — к области NAGET или AGET.

Зачем нужно такое разделение? Как правило, обычный текст, выводимый на экран, имеет постоянные атрибуты (например, желтые буквы на синем фоне). Неактивное поле ввода может иметь другие атрибуты, чтобы отличать его от обычного текста (например, белые буквы на синем фоне). И, наконец, активное поле ввода для привлечения вашего внимания должно иметь свои атрибуты (например, черные буквы на сером фоне). Такое различие в атрибутах позволит человеку легко ориентироваться среди информации на экране.

Нет нужды постоянно указывать при выводе желаемые атрибуты. С помощью функции setattr( ) можно установить постоянные атрибуты для каждой из трех областей. До нового вызова функции setattr( ) атрибуты будут оставаться неизменными.

Например, если в программу включить строку

```
setattr(SAY,vc.green + (vc.bg*vc.black));
```

то функции say( ), atsay( ) и т.п. будут после исполнения этой строки печатать текст зелеными буквами на черном фоне. Функция erase( ) очистит экран с черным цветом фона. Но функции xatsay( ) и xerase( ) по-прежнему позволят вам временно пользоваться другими атрибутами.

В заключение маленький пример. Создадим небольшой "электронный бланк". Бланк будет синего цвета. Весь "машинописный" текст будет белым. Поля, предназначенные для ввода данных, будут иметь желтые буквы. Поле, куда в данный момент будут вводиться данные, будет иметь черные символы на сером фоне.

```
Пример: #include <vcstdio.h>
        main( )
        {
        TEXT number[6];                 /* Номер детали */
        TEXT gost[8];                   /* ГОСТ */
        TEXT kod[6];                    /* код детали */
        TEXT data[9];                   /* дата выпуска */
        TEXT amount[7];                 /* количество деталей */
        COUNT key;

           vcstart(CLRSCRN);            /* инициализация VC */

        /* Установка атрибутов */
           setattr(SAY,vc.white + vc.bold + (vc.bg*vc.blue));
           setattr(NAGET,vc.brown + vc.bold + (vc.bg*vc.blue));
           setattr(AGET,vc.black + (vc.bg*vc.white));

        /* Инициализация переменных для полей "бланка" */
           empty(number,6);
           strcpy(gost,"1234-56");      /* ГОСТ по умолчанию */
           empty(kod,6);
           empty(data,8);
           strcpy(amount,"1");          /* кол-во по умолчанию */

        /* Начальное оформление экрана */
           erase( );                    /* очистить экран */
           atsay(1,25,"УЧЕТНАЯ КАРТОЧКА ИЗДЕЛИЯ");
           atsay(3,32,"Форма # 4");
           atsay(23,70,"7 отдел");

        /* Объявление полей ввода "бланка" */
           atsayget(7,5,"Серийный номер изделия: ",number,"99999");
           atsayget(12,5,"Код изделия: ",kod,"AXXXX");
           atsay(12,50,"Дата изготовления:");
           xatget(12,69,data,"99/99/99",valdate,
                          NULL,NULL,DEFAULT,DEFAULT);
           atsayget(17,5,"Общее количество: ",amount,"99999");

        /* Заполнение "бланка" */
           while (readgets( )! = RET);

           vcend(NOCLOSE);
        }
```

В этом примере при вызове функции xatget( ) в качестве функции, проверяющей корректность введенной даты, указана функция VC valdate( ). Эта функция не позволит вам ввести неправильную дату, например указать 29 февраля 1991 г.

ОКНА

До сих пор вы работали с целым экраном дисплея. Но можно выбрать участок экрана и объявить его окном (открыть окно). Тогда весь экранный ввод/вывод будет происходить только в этом окне, т.е. окно будет играть роль целого дисплея.

ОТКРЫТИЕ ОКНА

Проще всего открыть окно с помощью функции wopen( ).

Функция wopen( ):	c = wopen(upr,upc,lor,loc,title)		
	COUNT	c;	номер открытого окна (или -1 в случае неудачи
	COUNT	upr,upc;	координаты левого верхнего угла окна
	COUNT	lor,loc;	координаты правого нижнего угла окна
	TEXT	*title;	заголовок окна

```
Пример:   COUNT  wdo_a;
          wdo_a = wopen(10,10,20,30,"Это окно A");
```

После открытия окна все функции экранного ввода/вывода будут "считать", что окно — это и есть весь экран.

ЗАКРЫТИЕ ОКНА

Как вернуться в исходное состояние — отказаться от окна (закрыть окно)? Это делают с помощью функции WCLOSE( ).

Функция wclose( ):	c = wclose(number)		
	COUNT	c;	равно -1 в случае ошибки
	COUNT	number;	номер окна

ОДНОВРЕМЕННАЯ РАБОТА С НЕСКОЛЬКИМИ ОКНАМИ

Допустим, вы открыли несколько окон сразу. В какое из них будет выводиться информация, например, функцией say( ) и как вообще можно чередовать вывод в различные окна? Существует два основных решения этой проблемы.

Во-первых, в VC существует понятие "текущего окна". Идея текущего окна состоит в следующем. Пока нет открытых окон, функции экранного ввода/вывода — say( ), atsay( ), get( ) и прочие работают в пределах всего экрана дисплея. Как только открывается окно, функции начинают работать только в пределах этого окна. Если, имея одно открытое окно, открыть еще одно окно, функции экранного ввода/вывода будут работать только в последнем открытом окне. Окно, в котором работают функции экранного ввода/вывода, называется текущим окном. Обычно это окно, открытое последним по времени. Можно ли выводить не в последнее открытое окно? Да, можно. Функция wselect( ) позволяет сделать текущим любое из окон.

Функция wselect( ):	c = wselect(number)		
	COUNT	c;	
	COUNT	number;	номер окна

```
Пример:   #include <vcstdio.h>
          main( )
```

```
{
COUNT  wdo_1, wdo_2, wdo_3;

   vcstart(CLRSCRN);

   wdo_1 = wopen(10,10,20,20,"Окно 1");
   wdo_2 = wopen(5,30,12,70,"Окно 2");
   wdo_3 = wopen(14,25,19,50,"Окно 3");

   say("Вывод в окно #3"); /* wdo_3 - последнее открытое окно */
   wselect(wdo_1);
   say("Вывод в окно #1");
   wselect(wdo_2);
   say("Вывод в окно #2");
   getone( );
   vcend(CLOSE);
}
```

Во-вторых, вы можете явно указывать, в какое именно окно должна выводиться информация. Для этого применяются функции wsay( ), wat( ), watsay( ), wxatsay( ), werase( ) — это полные аналоги функций say( ), at( ), atsay( ), xatsay( ), xerase( ) соответственно с той лишь разницей, что на месте первого параметра при их вызове указывается номер окна, в который нужно направить вывод. Для экономии места здесь дается описание только функции wxatsay( ). Сравните его с описанием функции xatsay( ), и будет ясно, как должны выглядеть описания остальных функций.

Функция wxatsay( ):	VOID wxatsay(number,row,col,str,attr)		
	COUNT	number;	номер окна
	COUNT	row, col;	координаты начала строки
	TEXT	*str;	текст сообщения
	COUNT	attr;	атрибуты сообщения

Пример:
```
#include <vcstdio.h>
main( )
{
COUNT  wdo_1, wdo_2, wdo_3;

   vcstart(CLRSCRN);

   wdo_1 = wopen(10,10,14,20,"Окно 1"); /* открыть первое окно */
   wdo_2 = wopen(0,40,15,70,"Окно 2 "); /* открыть второе окно */
   wdo_3 = wopen(12,15,20,30,NULL);     /* открыть третье; оно  не
                                           имеет заголовка и частично
                                           перекрывает первое окно */
   getone( );                           /* пауза — нажмите
                                           любую клавишу */
   wsay(wdo_3,"Пишем в третье окно");
   getone( );
   watsay(wdo_2,2,24,"Пишем во второе окно");
   getone( );
   werase(wdo_1,vc.white*vc.bg);        /* закрасим первое окно
                                           белым цветом */
```

```
        getone( );
        wclose(wdo_3);                  /* закроем третье окно */
        getone( );                      /* теперь видна ранее
                                           закрытая часть
                                           второго окна */

        vcend(CLOSE);                   /* конец работы. VC сам
                                           закроет незакрытые
                                           окна wdo_1 и wdo_2 */

    }
```

АТРИБУТЫ ОКНА

Как вы уже могли заметить, каждое открытое с помощью функции wopen( ) окно имеет свои, отличные от других окон атрибуты границ, фона, текста и пр. Все они устанавливаются VC автоматически при каждом открытии окна без вашего участия. Атрибуты для окон берутся из таблицы цветов (она содержится в файле vcglobal.c). Эта таблица задает для каждого вновь открываемого окна свою комбинацию атрибутов. Этим комбинациям в VC присвоены порядковые номера. Следующая таблица показывает, какие комбинации атрибутов присваиваются открываемым окнам.

Порядковый номер открываемого окна	Номер комбинации атрибутов
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	0
11	1
12	2
13	3

Из таблицы видно, что после десятого открытия окна комбинации атрибутов начинают повторяться.

Можете ли вы сами выбирать атрибуты для окон по своему усмотрению? Да, можете. Для этого существует функция wattr( ).

Функция wattr( ):

```
c = wattr(wno, border, bground, say, actget, nactget,
                                    title, mode)
COUNT   c;
COUNT   wno;        номер окна
COUNT   *border;    указатель на атрибуты границы
COUNT   *bground;   указатель на атрибуты фона
COUNT   *say;       указатель на атрибуты для операций SAY
COUNT   *actget;    указатель на атрибуты для активных GET
```

```
COUNT    *nactget;   указатель на атрибуты для неактивных GET
COUNT    *title;     указатель на атрибуты заголовка
COUNT    mode;       признак операции:
                         SET — установить новые атрибуты
                         GET — получить текущие атрибуты
```

Вызов функции wattr( ) воздействует лишь на атрибуты той информации, которая выводится ПОСЛЕ этого вызова. Все, что было выведено ДО вызова функции wattr( ), останется без изменения.

Пример:
```
#include <vcstdio.h>
main( )
{
COUNT    wdo;
COUNT    w_border, w_bg, w_say, w_aget, w_naget, w_title;

    vcstart(CLRSCRN);

    wdo = wopen(10,10,20,20,"Окно");  /* открыть окно */
    say("Этот текст имеет атрибуты из таблицы цветов VC");
    wattr(wdo,&w_border,&w_bg,&w_say,&w_aget,&w_naget,
              &w_title,GET);          /* получить значения атрибутов */
    w_bg = vc.red*vc.bg;              /* изменить некоторые атрибуты */
    w_say = vc.blue + vc.bold + vc.blink + w_bg;
    wattr(wdo,&w_border,&w_bg,&w_say,&w_aget,&w_naget,
              &w_title,SET);          /* установить новые значения
                                         атрибутов */
    say("Этот текст имеет другие атрибуты");
                                      /* прежний текст остался без
                                         изменений */
    getone( );
    vcend(NOCLOSE);
}
```

ОТКРЫТИЕ ОКНА С ПОМОЩЬЮ ФУНКЦИЙ НИЗКОГО УРОВНЯ

Для каждого нового окна VC создает отдельную структуру специального типа WINF. Каждая такая структура полностью описывает одно из окон. И кроме границ и заголовка, окно характеризует еще множество других параметров. Однако при вызове функции wopen( ) мы этих параметров не указывали; в этом случае VC сам выбирал значения большинства параметров по своему усмотрению. Таким образом, функция wopen( ), с одной стороны, позволяет просто и быстро открыть окно, а с другой стороны, предоставляет очень мало возможностей для оформления окон по своему вкусу. Существует две функции более низкого уровня — wxopen( ) и wxxopen( ), — позволяющие создать окно практически на любой случай жизни. Естественно, вызов их производится сложнее, чем функции wopen( ).

```
Функция          c = wxopen(upr,upc,lor,loc,title,ctrl,dth,wth)
wxopen( ):       COUNT  c;          номер открытого окна (или -1 в случае
                                    неудачи )
                 COUNT  upr,upc;    координаты левого верхнего угла окна
```

```
                COUNT  lor,loc;     координаты правого нижнего угла окна
                TEXT   *title;      заголовок окна
                COUNT  ctrl;        управляющее слово окна
                COUNT  dth, wth;    ширина и высота виртуального экрана
```

Функция wxxopen( ):

```
c = wxxopen(upr,upc,lor,loc,title,ctrl,dth,wth,wc,fch)
COUNT  c;           номер открытого окна (или -1 в случае
                    неудачи)
COUNT  upr,upc;     координаты левого верхнего угла окна
COUNT  lor,loc;     координаты правого нижнего угла окна
TEXT   *title;      заголовок окна
COUNT  ctrl;        управляющее слово окна
COUNT  dth, wth;    ширина и высота виртуального экрана
COUNT  wc;          номер таблицы атрибутов цветов (или -1
                    для значения по умолчанию)
COUNT  fch;         код символа-заполнителя фона
```

Как видно из описаний, вызов этих функций требует задания множества новых параметров. Прежде всего, это управляющее слово окна. Это 16-битовая целая переменная, каждый бит которой соответствует наличию (или отсутствию) у окна какого-либо свойства. Обычно управляющее слово записывается не одним числовым значением, а в целях улучшения читаемости программы записывается как сумма значений:

значение1 + значение2 + значение3 + ...

Все значения, соответствующие битам управляющего слова окна, имеют в VC свои имена. Ниже приводится список этих значений с кратким пояснением их назначения.

BORDER	окно имеет рамку
BD1	двойная рамка
BD2	одинарная рамка
BD3	комбинированная рамка
ACTIVE	сразу рисовать окно на дисплее; если этот бит не присутствует в управляющем слове, то окно будет открыто, но на экране не будет видно
CURSOR	в окне будет виден курсор
SCROLL	производить прокрутку окна вверх по мере его заполнения
COOKED	управляющие коды (возврат каретки, перевод строки и пр.) будут отрабатываться при выводе их в окно, а не изображаться соответствующими символами
WINSERT	при выводе в окно будет работать режим вставки (аналогично режиму вставки для текстовых редакторов)
WDWRAP	при выводе в окно будет работать режим автопереноса (аналогично режиму автопереноса для текстовых редакторов)
SCROLLBARS	показывать текущую позицию в виртуальном окне (см. далее)
NOADJ	окно не подлежит дальнейшей модификации (см. далее)
STANDARD	выводить окно на экран сверху вниз
CENTER	выводить окно на экран из центра
TLDN	выводить окно на экран из левого верхнего угла

Пример открытия окна с одинарной рамкой:

```
wdo = (10,10,20,20,"Окно",ACTIVE + BORDER + BD2 + CENTER,0,0);
```

Окно будет "распахиваться" на экране дисплея от центра к краям (за это отвечает компонента CENTER).

Смысл параметров dth, wth и wc будет объяснен далее.

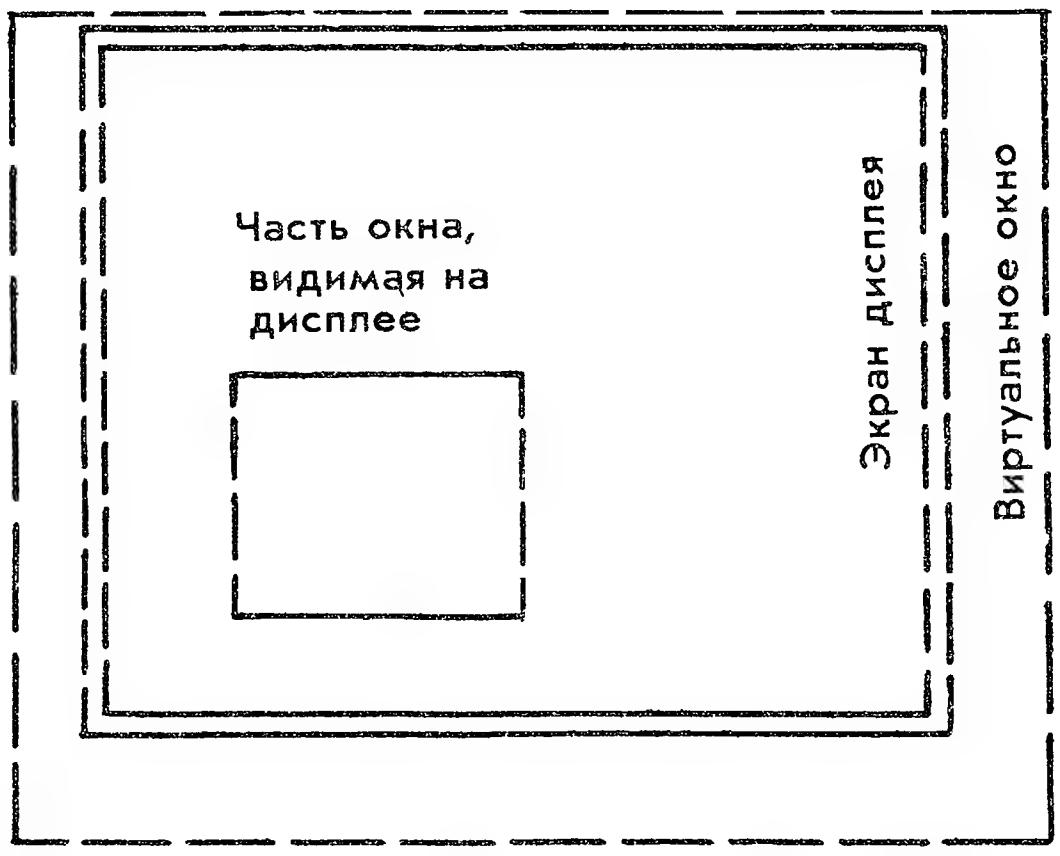
Код символа-заполнителя фона задает символ, которым будет заполнено

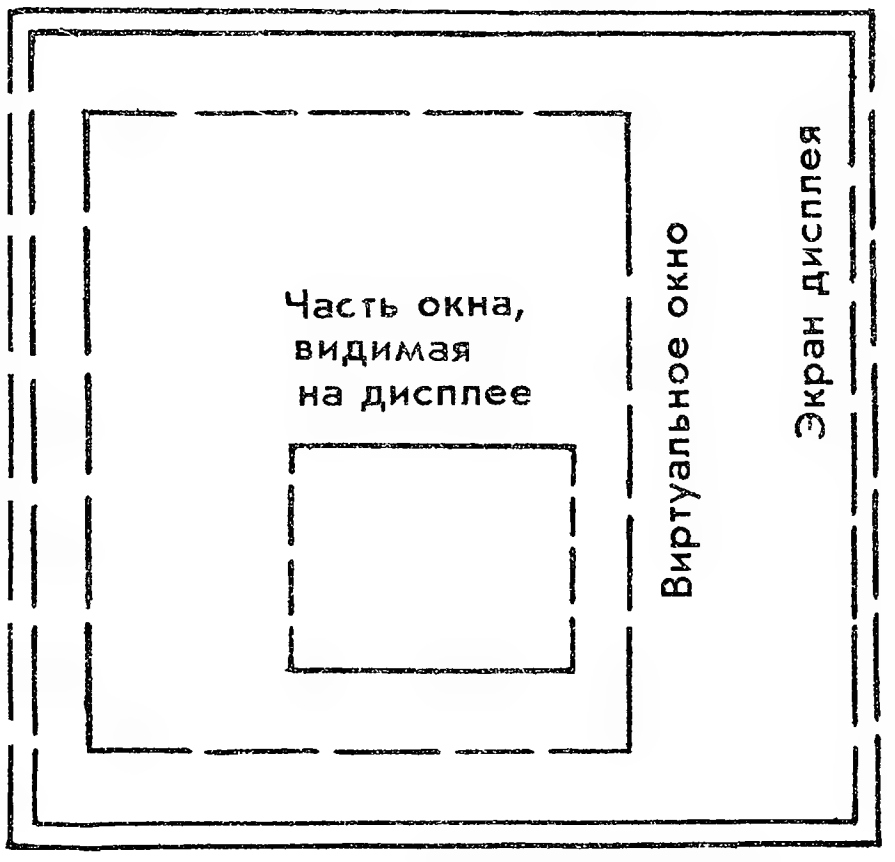
все окно после открытия. Чаще всего для заполнения фона используются символы с десятичными кодами 32 (пробел), 176, 177, 178.

ВИРТУАЛЬНЫЕ ОКНА

Что происходит, если при открытии окна его ширина и высота (параметры dth и wth — см. описания функций wxopen( ) и wxxopen( )) имеют значения большие, чем физические размеры, реально изображаемые на дисплее (параметры upr, upl, lor, loc)? Например, если открыто окно размером 10x20, а dth = wth = 50 ? В таком случае получается "виртуальное окно".

Виртуальное — потому что окно целиком не видно, и нам приходится как бы рассматривать его сквозь "замочную скважину", задаваемую параметрами upr, upl, lor и loc. Фактически получается одна из таких ситуаций:





Итак, с помощью функций wxopen( ) и wxxopen( ) можно открыть окно, реальные размеры которого больше, чем изображаемые на экране границами окна (и даже больше, чем размеры целого дисплея).

Возможно, что если значения параметров dth и wth окажутся меньше, чем задаваемые границами окна, тогда VC самостоятельно изменит их так, чтобы размеры виртуального окна в точности соответствовали изображению на экране. Поэтому если вы хотите открыть с помощью функций wxopen()/wxxopen() обычное (не виртуальное) окно, можете смело подставлять на месте параметров dth и wth значение 0.

МАНИПУЛЯЦИИ С ВИРТУАЛЬНЫМИ ОКНАМИ

Как перемещать видимую на экране "замочную скважину" по виртуальному окну? Это можно сделать несколькоми способами.

Во-первых, мы можем использовать функции, перемещающие курсор в ходе своей работы, at( ), wat( ), atsay( ), watsay( ) и др. Как только окажется, что курсор должен попасть в невидимую часть виртуального окна, наша "замочная скважина" перепрыгнет туда, и теперь эта часть окажется видимой.

Во-вторых, существует специальная функция wscroll( ).

Функция wscroll( ):	VOID wscroll(number,direction,amount)	
	COUNT number;	номер окна

COUNT	direction;	направление прокрутки, может принимать значения: CUP — прокрутка вверх CDOWN — прокрутка вниз CLEFT — прокрутка влево CRIGHT — прокрутка вправо
COUNT	amount;	число шагов прокрутки

Эта функция предназначена для "прокрутки" виртуального окна вправо/влево и вверх/вниз, подобно тому как обычно сдвигается вверх изображение на дисплее, когда оказывается заполненной самая нижняя строчка.

Пример:

```
/* Прокрутка виртуального окна вверх/вниз */
#include <vcstdio.h>
#define DEPTH 20
#define WIDTH 20
main( )
{
COUNT  wdo_1;
COUNT  i;
TEXT  c, line[80];

   vcstart(SAVESCRN);

   wdo_1 = wxopen(10,10,20,20,NULL,ACTIVE + SCROLLBARS + BORDER
             + BD2, DEPTH,WIDTH);    /* открыть окно */

/* Заполнить окно текстом */
   for (i = 0; i<DEPTH-2; i + + )
      { sprintf(line,"Строка #%d",i);
        atsay(i,1,line); }

   at(0,0);                                  /* перейти в начало окна */

/* Прокрутка при нажатии клавиш со стрелками вверх и вниз */
   while((c = getone( ))! = ESC)        /* выход — по нажатию ESC */
      switch(c)
      { case CUR_UP:        wscroll(wdo_1,CUP,1);
                            break;
        case CUR_DOWN:      wscroll(wdo_1,CDOWN,1);
                            break;
        default:            bell( );   /* нажата неверная клавиша */
                            break; }
   vcend(CLOSE);
}
```

При работе этой программы можно видеть, как в ходе прокрутки окна по левой границе окна движется черточка. Вся левая граница соответствует размеру виртуального окна по вертикали. Черточка показывает примерное расположение части виртуального окна, выводимой на экран.

МОДИФИКАЦИЯ ОКНА

VC предоставляет одну уникальную услугу. Если программа ожидает нажатия клавиши, то после нажатия функциональной клавиши <F2> у вас появ-

ляется возможность изменить размеры и положение текущего окна на экране. Внешне это выглядит так: вокруг окна появляется новая рамка, а внизу экрана — информационная строка, сообщающая, какими клавишами нужно пользоваться для модификации окна. Если модифицируемое окно ранее закрывало другие окна, то такая операция позволит сдвинуть это окно в другое место и сделать доступными для обозрения ранее скрытые части экрана. После модификации измененное окно останется таким, каким его сделали, до конца работы программы либо до новой модификации, т.е. до нового нажатия функциональной клавиши <F2>.

Обратите внимание на то, что для введения в программу такой довольно сложной возможности, как модификация окон, программисту не требуется писать ни строчки кода, необходимо лишь нажать одну клавишу. Все остальное сделает сам VC!

Возможность последующей модификации каждого из окон задается все в том же управляющем слове окна при его открытии. Обычно эта возможность включена. Если по каким-то причинам необходимо отказаться от этой услуги, к конструкции, формирующей управляющее слово окна, необходимо прибавить величину NOADJ.

```
Пример:  wdo_1 = wxopen(10,10,20,20,NULL,ACTIVE + BORDER + BD2,0,0);
                                    /* окно можно впоследствии модифицировать */
         wdo_2 = wxopen(10,10,20,20,NULL,ACTIVE + BORDER + BD2 + NOADJ,
         0,0);
                                    /* окно нельзя впоследствии модифицировать */
```

Модификацию окон можно производить и без нажатия функциональной клавиши <F2>. Если в программе есть вызов функции wadjust( ) — результат будет тот же. Строго говоря, при нажатии функциональной клавиши <F2>

```
Функция          c = wadjust(number);
wadjust( ):      COUNT    c;
                 COUNT    number;      номер окна
```

VC просто-напросто исполняет функцию wadjust( ) для текущего окна.

НЕКОТОРЫЕ ПОЛЕЗНЫЕ ВОЗМОЖНОСТИ ПРИ РАБОТЕ С ОКНАМИ

Если необходимо на некоторое время убрать окно с экрана, а затем вновь отобразить его, можно воспользоваться функциями whide( ) и wshow( ).

```
Функция          c = whide(number);
whide( ):        COUNT    c;          в случае ошибки равно -1
                 COUNT    number;     номер окна

Функция          c = wshow(number);
wshow( ):        COUNT    c;          в случае ошибки или если окно
                                      уже отображено на экране, равно -1
                 COUNT    number;     номер окна
```

Функция whide( ) "гасит" указанное окно; но само окно продолжает существовать. Все, что было изображено в этом окне, не теряется, а сохраняется в оперативной памяти в образе окна. Более того, в такое "погашенное" окно

можно выводить информацию, и эта информация будет помещена в невидимый образ окна. Когда потребуется отобразить на экране дисплея погашенное окно, достаточно выполнить вызов функции wshow( ), и на экране вновь появится погашенное окно со всем содержимым, которое было в окне раньше, плюс та информация, которую, возможно, выводили в окно, когда оно было погашено.

Пример:
```
#include <vcstdio.h>
main( )
{
COUNT   wdo;

   vcstart(CLRSCRN);
   wdo = wopen(10,10,20,20,"Окно");
   atsay(1,3,"Первая запись");
   getone( )
   whide(wdo);
   atsay(2,4."Вторая запись  —  в погашенное окно");
   getone( );
   wshow(wdo);
   getone( );
   vcend(CLOSE);
}
```

Если вас не устраивает внешний вид открытого окна, можно воспользоваться функцией wattr( ) (см. раздел АТРИБУТЫ ОКНА). Но иногда желательно не открывать окно и затем менять его атрибуты, а сразу получать на экране окончательное изображение окна. В таких случаях при открытии окна путем вызова функций wxopen( ) и wxxopen( ) не указывайте компоненту ACTIVE в управляющем слове окна. Тогда окно откроется, но изначально будет невидимо в погашенном состоянии (см. функцию whide( )).

Теперь можно делать с окном что угодно — до вызова функции wshow( ) никто манипуляций с окном на экране не увидит.

Пример:
```
#include <vcstdio.h>
main( )
{
COUNT  wdo;

   vcstart(CLRSCRN);

   wdo = wxxopen(10,10,20,20,"Окно",CENTER + NOADJ + BORDER +
                          BD1,0,0,0,176);
                          /* окно открыто, но пока невидимо  —  в уп-
                          равляющем слове нет компоненты ACTIVE */

   w_bg = vc.black*vc.bg;
   w_border = vc.black + w_bg
   w_title = vc.red + vc.bold + w_bg;
   w_say = vc.black*vc.bg;
   w_aget = vc.red + vc.bold + w_bg;
   w_naget = vc.red + w_bg;
   wattr(wdo,&w_border,&w_bg,&w_say,&w_aget,&w_naget,
               &w_title,SET);      /* установить нужные атрибуты */
```

```
    atsay(1,5,"Строка символов")

/* До сих пор на экране ничего не отображалось */

    whow(wdo_i);                    /* теперь окно появилось на экране */

    getone( );
    vcend(CLOSE);
}
```

Вот еще одно полезное свойство. После открытия хотя бы одного окна курсор все время будет находиться в пределах окна. Все, что мы пока умеем, — это перемещать курсор из одного окна в другое с помощью функции wselect( ). А можно ли поместить курсор или выводить информацию в любое место дисплея? Да, можно. Это делается при помощи переменной VC override. Если эта переменная имеет значение 0, то курсор можно перемещать только внутри окон. Если переменная override имеет ненулевое значение, то курсор можно поместить куда угодно.

Пример:
```
wdo = wopen(10,10,20,20,"Окно");
atsay(3,2,"Строка выводится в окно");
override = 1;
atsy(3,2,"Строка выводится мимо окна");
override = 0;
atsay(4,2,"Строка снова выводится в окно");
```

Иногда вывод информации производится в далеко отстоящие друг от друга части виртуального окна. В таком случае изображение неприятно прыгает туда-сюда. Этого можно избежать, воспользовавшись функциями woff( ) и won( ).

Функция woff( ): VOID woff( );

Функция won( ): VOID won( );

Вызов функции woff( ) как бы "замораживает" изображение окна на экране. Окно остается в неизменном виде, что бы в него не выводилось. После вызова функции won( ) окно перерисовывается, и вы практически мгновенно получаете обновленное изображение заполненного информацией окна.

ЦИКЛИЧЕСКИЕ ФУНКЦИИ

Хотите иметь в своей программе "встроенные часы", отображающие текущее время? Это очень просто. Достаточно включить в программу строки

```
sw = wxopen(24,72,24,79,NULL,ACTIVE,0,0);
timeloc(sw,0,0,DEFAULT);
setloop(timeloop);
getone( );                          /* пауза — полюбуйтесь на часы */
```

и в области экрана с координатами (24,72) будет отображаться текущее время.

```
Функция         VOID timeloc(wno,row,col,attr)
timeloc( ):     COUNT  wno;     номер окна для вывода текущего времени
                COUNT  row,col; координаты для вывода в этом окне
                COUNT  attr;    атрибуты для вывода (или DEFAULT)

Функция         COUNT timeloop( )
```

Как устроены такие часы? Маленькая хитрость. Любой запрос VC на ввод с клавиатуры (readgets( ) или другая функция) в конце концов приводит к вызову функции getone( ). Эта базовая функция ждет ввода символа, а пока символ еще не введен, может выполнить какое-либо несложное действие (например, вывести на дисплей текущее время). После этого getone( ) снова проверяет, введен ли символ. И снова, если символ еще не введен, выполняет то же несложное действие, и т.д.

МЕНЮ

Конечно же, VC позволяет вам создавать разнообразные меню. Меню в VC бывают:

- горизонтальные;
- вертикальные.

Меню содержат пункты. Пункты могут быть:

- пунктами, которые могут быть выбраны (на них можно остановить курсор);
- пунктами, которые не могут быть выбраны (через них курсор перескакивает без остановки);
- пустыми (просто строками пробелов для улучшения наглядности меню);
- разделителями (горизонтальными или вертикальными линиями в зависимости от вида меню для улучшения наглядности меню);

Для выбора пункта меню необходимо установить на него курсор и нажать клавишу <Enter> (<ВК>). При выборе пункта меню может происходить:

- вызов указанной функции;
- вызов другого меню.

Общий порядок действий при создании и использовании меню следующий.

1. С помощью функции addvcmstyle( ) создается [новый] тип меню.
2. С помощью функции menunew( ) создается меню этого типа.
3. С помощью функции menuitem( ) созданное меню заполняется пунктами.
4. Вызовом функции vcmenu( ) меню вызывается к работе.
5. После нажатия клавиши <Esc> меню заканчивает работы.

```
Функция          c = addvcmstyle(name,mnustyle,width,norm,bar,
addvcmstyle( ):  unav,bord,permlvl);
                 COUNT    c;
                 TEXT     *name;      Символическое имя типа меню
                 COUNT    mnustyle;   Опции типа меню. Могут иметь значения:
                                        HORIZONTAL  горизонтальное меню
                                        VERTICAL    вертикальное меню
                                        BORDERLESS  меню без границ
                                        TITLELEFT   заголовок печатать слева
                                        TITLERIGHT  заголовок печатать справа
```

			TITLECENTER заголовок печатать в центре
	COUNT	width;	Ширина окна меню
	COUNT	norm;	Атрибуты для нормальных пунктов меню
	COUNT	bar;	Атрибуты для подсвеченного прямоугольника
	COUNT	unav;	Атрибуты для недоступных пунктов
	COUNT	bord;	Атрибуты для границы меню
	COUNT	permlvl;	Уровень доступности меню
Функция menunew( ):	m = menunew(row,col,mnunam,hlpkey,style);		
	VCMENU	m;	
	COUNT	row,col;	верхняя строка и левая колонка
	TEXT	*mnunam;	заглавие меню
	TEXT	*hlpkey;	указатель на контекстное вспомогательное сообщение
	TEXT	*style;	имя типа меню Vitamin C
Функция menuitem( ):	m = menuitem(menuptr,item,func,parm,stat,ctrl);		
	MENUITEM	*m;	Указатель на пункт меню, если нет ошибок, или NULL
	VCMENU	*menuptr;	Указатель на требуемое меню
	TEXT	*item;	Имя пункта для появления в меню
	PFI	func;	Функция для вызова, если выбран, или NULLFUNC
	TEXT	*parm;	Указатель на параметры
	TEXT	*stat;	Указатель на строку статус-сообщения
	COUNT	ctrl;	Опции пункта меню. Могут иметь значения:
			UNAVAILABLE пункт не может быть выбран
			SEPARATOR разделитель
			MENU если пункт выбран, переход к меню
			RETURN закрыть родительское меню после прекращения работы меню
			HIDE прятать меню на время выполнения функции
Функция vcmenu( ):	c = vcmenu(m)		
	COUNT	c;	0, если ошибка, иначе не 0
	VCMENU	*m;	Указатель на требуемое меню

Пример построения и использования меню:

```
#include <vcstdio.h>
extern int func1(void);
extern int func2(void);
extern int func3(void);

main( )
        {
        VCMENU *menunew( );
        VCMENU *mainmenu;
        MENUITEM *menuxitem( );
        COUNT func1( ), func2( ), func3( );
```

```
vcstart(CLRSCRN);

addvcmstyle("STDHORIZ",HORIZONTAL | TITLECENTER,80,
            vc.cyan + vc.bold + vc.bg*vc.blue,   /* нормальный */
            vc.blue + vc.bg*vc.white,            /* выделенный */
            vc.cyan + vc.bg*vc.blue,             /* не может быть
                                                    выбран */
            vc.white + vc.blue*vc.bg,            /* границы   */
            0);
mainmenu = menunew(0,0,"Vitamin C — Пример меню",
"NO HELP","STDHORIZ");
menuitem(mainmenu,"Первый пункт",
func1,NULL,NULL,STRPARM);
menuitem(mainmenu,"Второй пункт",
func2,NULL,NULL,STRPARM);
menuitem(mainmenu,"Третий пункт",
func3,NULL,NULL,STRPARM);

vcmenu(mainmenu);

vcend(NOCLOSE);
}
```

ОСНОВНЫЕ НЕДОСТАТКИ ПАКЕТА

Естественно, VC присущи и некоторые недостатки. Так, первое, что бросается в глаза, после того, как программа скомпилирована и собрана, это ее неожиданно большие размеры. Функции VC, подключаемые к программе на этапе сборки, существенно увеличивают ее объем. Минимальные размеры программы, использующей VC, составляют около 20 Кбайт, даже если программа содержит всего 3-4 строки. Размеры программы по мере увеличения длины ее исходного текста довольно быстро достигают примерно 60 Кбайт, после чего рост длины заметно тормозится и после достижения около 80 Кбайт (для программ с весьма длинным исходным текстом) далее практически уже не заметен. К этой особенности VC следует быть готовым.

Второй существенный недостаток пакета — то, что при работе его функции требуют довольно много оперативной памяти. Поэтому вероятно, что достаточно большие программы вам придется писать и компилировать для большой (large) модели памяти.

Третий недостаток пакета — это его невосприимчивость к русским буквам. Попытка ввести русский текст скорее всего будет неудачной (правда, может быть именно ваша версия VC будет свободна от этого недостатка). Проще всего решить эту проблему можно, заменив в файле vcstdio.h строку

```
#define FBIT 256
```

на строку

```
#define FBIT 512 .
```

Еще один крупный недостаток — то, что VC умеет вводить и выводить только символьную информацию. А как же, например, печатать числа? Это приходится делать в два приема. Сначала стандартной функцией sprintf( ) числа преобразуются в строки, а затем с помощью функций экранного ввода/вывода VC выводятся на экран.

```
Пример:     #include <vcstdio.h>
            main( )
            {
            int       number;
            TEXT      line[81];

               vcstart(CLRSCRN);

               number = 123;
               sprintf(line,"Результат: %3d",number);
               atsay(10,15,line);

               vcend(NOCLOSE);
            }
```

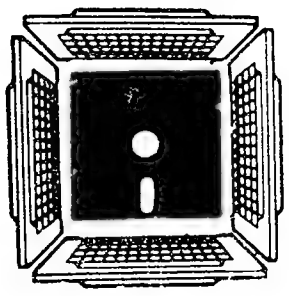
Ввод числовой информации производится сходным образом. Сначала вводится символьная строка Затем с помощью стандартной функции sscanf( ) из нее извлекаются числовые величины.

Прочие недостатки, по нашему мнению, являются не принципиальными и достаточно легко преодолимыми.

МОЖНО ЛИ ИСПОЛЬЗОВАТЬ НЕСКОЛЬКО ПАКЕТОВ ЧЕЛОВЕКО-МАШИННОГО ИНТЕРФЕЙСА СРАЗУ?

Допустим, вы познакомились с несколькими пакетами типа VC, оценили их достоинства и недостатки. Возникает вопрос: "Можно ли "взять" работу с окнами из пакета Window Professional, а систему меню — из пакета C-Scape"? Общий ответ — НЕТ. Почему? Вспомним раздел "Основная концепция". Все программные продукты типа VC работают сходным образом. Каждый пакет требует, чтобы ВЕСЬ ввод/вывод производился ТОЛЬКО СРЕДСТВАМИ ЭТОГО ПАКЕТА. Любые другие средства — стандартные, ваши собственные (не опирающиеся на средства этого пакета) запрещаются. При соблюдении этого условия пакет всегда будет точно "знать", что, где и как выводилось на экран. Если вы потребуете удалить какое-то окно, ваш пакет будет "знать", нужно ли восстанавливать область, закрытую этим окном, изменялась ли информация закрытой области и т.п. Если применить "постороннюю" функцию экранного вывода, ваш пакет ее "не заметит". Не заметит, если эта функция затрет часть экрана; не заметит, если поверх "постороннего" вывода сам начнет что-либо писать.

Поэтому помните: в пределах одной программы можно использовать ТОЛЬКО ОДИН пакет человеко-машинного интерфейса — только Vitamin C, или только Window Professional, или только C-Scape...



Л.Н.Жариков,
Ф.Д.Сухоруков

РУСИФИКАЦИЯ ПРИНТЕРА LC15 Star (Япония)

Одной из первых проблем, встающих перед пользователями персональных компьютеров типа IBM PC XT/AT, является русификация принтера. Наиболее надежный способ для этого — замена микросхемы постоянного запоминающего устройства, в которой прошиты таблицы национальных шрифтов, но этот способ многим не по карману и требует поиска организации, которая бы делала такую работу. При другом способе русификации нужный шрифт создается специальной программой.

Многие известные прикладные системы, ориентированные на работу с текстами, дают возможность работать с русским шрифтом и выводить тексты на печать в графическом режиме практически на любой матричный принтер. Такими системами являются, например, LETTRIX и LEXICON. Тексты получаются достаточно высокого качества, но выводятся на печать настолько медленно, что практически непригодны для использования при отладке, когда требуется быстро получить текущую копию текста программы.

Часто предлагается другой способ русификации принтера, с помощью которого можно получить быстрый точечный шрифт. Большинство современных матричных принтеров имеют собственный процессор, постоянное и оперативное запоминающие устройства, что позволяет, в частности, делать загрузку точечных национальных шрифтов во вторую половину кодовой таблицы символов. Этим способом мы и воспользовались.

Вниманию читателей предлагается программа, написанная на языке Турбо-ПАСКАЛЬ, в какой-то мере помогающая решить эту задачу для девятиигольчатых принтеров типа Epson. Программа проверена на принтере японского производства LC15 Star.

Чтобы получить точечный шрифт, были закодированы все строчные и заглавные буквы русского алфавита. Принципы кодирования шрифта изложены, например, в книге Брябрина В.М. "Программное обеспечение персональных ЭВМ" (М., Наука, 1989).

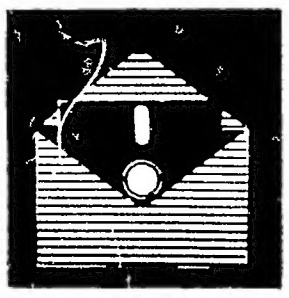
Вся информация о каждом символе находится в массиве S (см. текст программы). Программа достаточно проста. Необходимые пояснения приведены в комментариях. Перед использованием данной программы необходимо настроить принтер и откомпилировать ее в системе Турбо-ПАСКАЛЬ. Вся настройка принтера заключается в установке режима использования загруженного шрифта и выборе одной из альтернативных кодовых таблиц переключателями, которые установлены на пульте управления принтером.

Функциональное назначение каждого переключателя описано в руководстве, прилагаемом к принтеру.

При внесении некоторых изменений в программу она может быть использована для русификации других типов принтеров.

```
Program LC15Rus;
uses
  Printer;
const
  S : array[1..768] of byte =
{А} (139, 0, 30, 32, 72, 128, 8, 128, 72, 32, 30, 0,
{Б}  139, 0,254, 0,146, 0,146, 0,146, 0,140, 0,
{В}  139, 0,254, 0,146, 0,146, 0,146, 0,108, 0,
{Г}  139, 0,254, 0,128, 0,128, 0,128, 0,128, 0,
{Д}  139, 3, 0, 62, 64, 130, 0,130, 0,254, 0, 3,
{Е}  139, 0,254, 0,146, 0,146, 0,146, 0,130, 0,
{Ж}  139, 130, 68, 40, 16, 0,254, 0, 16, 40, 68,130,
{З}  139, 0, 68, 0,130, 0,146, 0,146, 0,108, 0
{И}  139, 0,254, 0, 4, 0, 8, 0, 16, 0,254, 0,
{Й}  139, 0,254, 0, 4, 0,200, 0, 16, 0,254, 0,
{К}  139, 0,254, 0, 16, 0, 16, 40, 68,130, 0,
{Л}  139, 0, 2, 0, 2,124,128, 0,128, 0,245, 0,
{М}  139, 0,254, 0, 64, 32, 16, 32, 64, 0,254, 0,
{Н}  139, 0,254, 0, 16, 0, 16, 0, 16, 0,254, 0,
{О}  139, 0,124, 130, 0,130, 0,130, 0,130,124, 0,
{П}  139, 0,254, 0,128, 0,128, 0,128, 0,254, 0,
{Р}  139, 0,254, 0,144, 0,144, 0,144, 0,96, 0,
{С}  139, 0,124, 0,130, 0,130, 0,130, 0,68, 0,
{Т}  139, 0,128, 0,128, 0,254, 0,128, 0,128, 0,
{У}  139, 0,228, 0, 18, 0, 18, 0, 18, 0,252, 0,
{Ф}  139, 0, 56, 0, 68, 0,254, 0, 68, 0, 56, 0,
{Х}  139, 0,198, 0, 40, 0, 16, 0, 40, 0,198, 0,
{Ц}  139, 0,254, 0, 2, 0, 2, 0, 2, 252, 2, 1,
{Ч}  139, 0,224, 0, 16, 0, 16, 0, 16, 0,  254, 0,
{Ш}  139, 0,254, 0, 2, 0,254, 0, 2, 0,254, 0,
{Щ}  139, 0,254, 0, 2, 0,254, 0, 2, 0,254, 1,
{Ъ}  139,128, 0,254, 0, 18, 0, 18, 0, 18, 0, 12,
{Ы}  139, 0,254, 0, 18, 0, 18, 12, 0, 0,254, 0,
{Ь}  139, 0,254, 0, 18, 0, 18, 0, 18, 0, 12, 0,
{Э}  139, 0, 68, 0,146, 0,146, 0,146, 68, 56, 0,
{Ю}  139, 0,254, 0, 16, 0,124, 130, 0,130,124, 0,
{Я}  139, 0, 98, 4,152, 0,144, 0,144, 0,254, 0,
{а}  139, 0, 4, 0, 42, 0, 42, 0, 42, 0, 28, 2,
{б}  139, 0, 28, 32, 82, 0, 82, 0, 82, 0,140, 0,
{в}  139, 0, 62, 0, 42, 0, 42, 0, 42, 0, 20, 0,
{г}  139, 0, 62, 0, 32, 0, 32, 0, 32, 0, 32, 0,
{д}  139, 0, 3, 0, 30, 32, 2, 32, 2, 60, 3, 0,
{е}  139, 0, 28, 34, 8, 34, 8, 34, 8, 34, 24, 0,
{ж}  139, 0, 34, 20, 8, 0, 62, 0, 8, 20, 34, 0,
{з}  139, 0, 20, 0, 34, 0, 42, 0, 42, 0, 20, 0,
{и}  139, 0, 62, 0, 4, 0, 8, 0, 16, 0, 62, 0,
{й}  139, 0, 62, 0, 4, 64, 8, 64, 16, 0, 62, 0,
{к}  139, 0, 62, 0, 8, 0, 8, 0, 20, 0, 34, 0,
{л}  139, 0, 2, 0, 2, 28, 32, 0, 32, 0, 62, 0,
{м}  139, 0, 62, 0, 16, 8, 4, 8, 16, 0, 62, 0,
{н}  139, 0, 62, 0, 8, 0, 8, 0, 8, 0, 62, 0,
{о}  139, 0, 28, 34, 0, 34, 0, 34, 0, 34, 28, 0,
{п}  139, 0, 62, 0, 32, 0, 32, 0, 32, 0, 62 0,
{р}  27, 0,127, 0, 68, 0, 68, 0, 68, 0, 56, 0,
{с}  139, 0, 28, 0, 34, 0, 34, 0, 34, 0, 34, 0,
{т}  139, 0, 32, 0, 32, 0, 62, 0, 32, 0, 32, 0,
{у}  27, 0, 65, 32, 17, 10, 4, 8, 16, 32, 64, 0,
{ф}  27, 0, 56, 0, 68, 0,255, 0, 68, 0, 56, 0,
{х}  139, 0, 34, 0, 34, 20, 8, 20, 34, 0, 34, 0,
{ц}  139, 0, 62, 0, 2, 0, 2, 0, 2, 0, 62, 1,
{ч}  139, 0, 48, 0, 8, 0, 8, 0, 8, 0, 62, 0,
{ш}  139, 0, 62, 0, 2, 0, 62, 0, 2, 0, 62, 0,
{щ}  139, 0, 62, 0, 2, 0, 62, 0, 2, 0, 62, 1,
{ъ}  139, 32, 0, 62, 0, 10, 0, 10, 0, 10, 0, 4,
{ы}  139, 0, 62, 0, 10, 0, 10, 0, 4, 0, 62, 0,
{ь}  139, 0, 62, 0, 10, 0, 10, 0, 10, 0, 4, 0,
{э}  139, 0, 20, 34, 0, 34, 8, 34, 8, 34, 28, 0,
{ю}  139, 0, 62, 0, 8, 0, 28, 34, 0, 34, 28, 0,
{я}  139, 0, 18, 0, 44, 0, 40, 0, 40, 0, 62, 0 );
var
  Esc : char;
Procedure LoadDraft (M, N, L : byte);
var i,j  : byte;
  Symbol : char;
begin
 for i: = M to N do
  begin
   Symbol: =Chr(L+i);

    {Установка режима загрузки точечного шрифта}
   Write(lst,Esc, '&', Chr(0),Symbol,Symbol);

    {Загрузка символа}
   for j:=i*12-11 to i*12 do Write(lst,Chr(S[j]));

    {Инициализация символа}
   Write(lst,Esc, '%', Chr(1));
  end;
end; { LoadDraft }

begin
 Esc:=Chr(27);
 {Установка межстрочного интервала 13/216 дюйма }
 Write (lst,Esc, 'A', Chr(13),Chr(10));

 [Загрузка русского шрифта в принтер и его инициализация }
 LoadDraft(1,48,127);
 LoadDraft(49,64,175);
 LoadDraft(49,49,63);
end.
```

Нам пишут



Я по образованию математик, постоянно занимаюсь разработкой программного обеспечения для решения научно-технических задач. Пока задачи были небольшими, не было и проблем с отладкой программ. Но стоило влезть в метод конечных элементов — и пришлось осваивать не только сам метод, но и технологию программирования. В данной заметке излагается один из найденных автором приемов программирования.

Г.Н.Гутман

Метод спящей отладки, или Как достучаться к подпрограмме?

Представьте себе, что вы — разработчик достаточно большой программы, состоящей из 20-30 и более отдельных модулей. После длительного и мучительного процесса тестирования и отладки ваша программа наконец-то работает. И вот, "очистив" ее от всевозможных отладочных операторов, в основном печатающих промежуточные результаты, вы обнаруживаете, что при какой-то комбинации входных данных программа снова "не идет". Увы, в каждой работающей программе есть по крайней мере одна ошибка! И чаще всего ее локализация и устранение требуют введения в текст программы тех же отладочных операторов, которые вы с легким сердцем из нее выбросили.

В данной заметке автор предлагает использовать разработанный им метод, который можно назвать методом спящей отладки.

Давайте сначала немного помечтаем. Хорошо бы, чтобы отладочный оператор печати, попав один раз в подпрограмму, оставался бы там после отладки в "спящем" состоянии и мог бы "проснуться" по вашему желанию. Тогда, заподозрив программу в неверности, можно было бы, включая по очереди или сразу все отладочные печати, снова проследить в деталях ход выполнения вашей программы.

Сразу возникают два вопроса. Первый: "Как сообщить программе о вашем желании?" Второй: "Как "достучаться" к нужной подпрограмме?" В принципиальном плане ответ на оба вопроса один — через входные данные. Во входных данных перечисляются имена подпрограмм, в которых вы хотите включить отладочные печати. Встроенный в программу отладчик прочитает эти имена и запишет их в список имен отлаживаемых подпрограмм. Каждая отладочная печать представляет собой вызов одного из модулей встроенного отладчика. В качестве фактических параметров передается имя подпрограммы, из которой сделан вызов модуля отладчика, и список переменных или массивов, содержимое которых должно быть напечатано. Подпрограмма печати ищет переданный ей идентификатор в списке имен отлаживаемых подпрограмм. И если находит — печатает все, что у нее попросили.

Очевидно, что конкретная реализация отладчика зависит от многих факторов, в том числе от используемого языка и транслятора. В частности, автор разработал встроенный отладчик на до сих пор еще широко используемом ФОРТРАНе. Для ознакомления с текстом и руководством по использованию встроенного отладчика обращаться непосредственно к автору по адресу: 443026, г.Куйбышев, Краснoглинское шоссе, д. 27, кв. 22.

В.Шохан,
студент Минского радиотехнического института

Хитрости форматирования

Многие пользователи советских персоналок ЕС 1840, ЕС1841 если и подозревают, но не знают, что дискеты для этих компьютеров можно форматировать на 720 Кбайт. Т.е. на дискете вместо двух сторон по 360 Кбайт будет одна на 720 К. Преимущества: не надо вспоминать, на какой стороне диска, что записано, можно записывать программы занимающие более 360К, и т.д. Такое форматирование можно делать различными способами. Для тех, кто пользуется операционной системой MS. DOS 3.3 это лучше всего делать с помощью системных внешних команд DM8 и FORMAT. Для этого с командной строки вводится:

DM8/F/Q x:

где x: - это сторона которую надо отформатировать (например a: или b:)

ЛЕНИНГРАДСКИЙ ПРОИЗВОДСТВЕННЫЙ КООПЕРАТИВ "ОСТ"

ПРЕДЛАГАЕТ

- программное обеспечение для пользователей компьютеров БК-0010,0010.01, ДВК-2М,3М, УКНЦ классов информатики КУВТ-86, КУВТ-УКНЦ.

Кооператив располагает обширным банком игровых, учебных, системных, прикладных программ.

Расценки значительно ниже государственных.

Списки программ при указании типа ЭВМ высылаются бесплатно.

ДЛЯ ПОЛЬЗОВАТЕЛЕЙ

БК-0010,0010.01:

- программную разработку, позволяющую на каждой кассете пользователя размещать в три раза больше программ по сравнению с традиционным способом записи, при многократном увеличении надежности (магнитофон любого типа);

- недорогой телефонный модем на базе БК-0010, 0010.01.

Адрес для переписки: 189510, Ленинград, а/я 649.

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР ПСМО "Северовостокэнергострой"

реализует интеллектуальные терминалы ЕС-8534.03 (см. статью в сборнике "Вычислительная техника и ее применение". — 1989. — № 12. — С.14), представляющие собой микро-ЭВМ, способные работать как ПЭВМ с возможностью передачи данных по коммутируемым каналам телефонной сети общего пользования и ПД-200.

Обращаться по адресу: 681015, Комсомольск-на-Амуре, ул.Кирова, 16.

Телефон: 4-89-21.

МОЛОДЕЖНЫЙ КОМПЬЮТЕРНЫЙ ЦЕНТР "КОНТАКТ"

предлагает пользователям персональных ЭВМ PC, Sinclair, Commodore-64, Atari, Yamaha, Радио-86РК, Микроша, Криста, Партнер, Львов, Специалист, Вектор, БК-0010/0010.01/0011, УКНЦ, Корвет, ДВК-2/-3 широкий выбор программ и документации.

Наш адрес: 614010, г.Пермь, а/я 4694.

П53 **Пользователям профессионального компьютера.** — М.: Знание, 1990. — 48 с. — (Новое в жизни, науке, технике. Сер. "Вычислительная техника и ее применение"; № 11).

ISBN 5-07-001610-5

20 к.

В сборнике приведены статьи по NORTON COMMANDER и по пакету VITAMIN, предназначенных для профессиональных ПЭВМ.

Материал рассчитан на широкий круг читателей.

ТЕМА *СЛЕДУЮЩЕГО* **НОМЕРА:**

РАДИО-ЭЛЕКТРОНИКА И СВЯЗЬ

СИСТЕМЫ КОЛЛЕКТИВНОГО ПОЛЬЗОВАНИЯ

***ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА* И ЕЁ ПРИМЕНЕНИЕ**

АВТОМАТИЗИРОВАННЫЕ ХАРАКТЕРИСТИКИ

МАТЕМАТИКА КИБЕРНЕТИКА

ДЖОН ФОН НЕЙМАН

Научно-популярное издание

ПОЛЬЗОВАТЕЛЯМ ПРОФЕССИОНАЛЬНОГО КОМПЬЮТЕРА

Гл.отраслевой редактор *Г. Г. Карвовский*
Редактор *Б. М. Васильев*
Мл.редактор *Н. А. Васильева*
Художник *В. Н. Конюхов*
Худож.редактор *И. А. Емельянова*
Техн.редактор *Т. В. Луговская*
Корректор *В. И. Гуляева*

ИБ № 11546

Подписано к печати 28.10.90. Формат бумаги 70x100 1/16. Бумага офсетная. Печать офсетная. Усл.печ.л.3,90. Усл.кр.-отт.8,45. Уч.-изд.л.3,69. Тираж 71574 экз. Заказ 1715. Цена 20 коп. Издательство "Знание". 101835, ГСП, Москва, Центр, проезд Серова,д.4. Индекс заказа 904711. Отпечатано с оригинал-макета издательства "Знание" на ордена Трудового Красного Знамени Тверском полиграфическом комбинате Государственного комитета СССР по печати. 170024, г.Тверь, пр.Ленина,5.

Цена 20 коп.

Индекс 70195

Адрес подписчика:

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

И ЕЕ ПРИМЕНЕНИЕ

. . .Так как только человек может знать, что ему нужно, то именно поэтому он будет ставить перед машинами задачи и направлять их на путь решения тех или иных проблем.

В.М. Глушков

Наш адрес:
СССР,
Москва,
Центр,
проезд
Серова, 4